



ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - TE141599

**PERANCANGAN SISTEM STABILISASI KAMERA TIGA
SUMBU DENGAN METODE KONTROL FUZZY UNTUK
*MOBILE SURVEILLANCE ROBOT***

Fahrezi Alwi Muhammad
NRP 2212 100 150

Dosen Pembimbing
Dr. Muhammad Rivai, S.T., M.T.
Suwito, S.T., M.T.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016



ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - TE141599

**DESIGN OF THREE AXIS CAMERA STABILIZER WITH
FUZZY CONTROL METHOD FOR MOBILE
SURVEILLANCE ROBOT**

Fahrezi Alwi Muhammad
NRP 2212 100 150

Advisor
Dr. Muhammad Rivai, S.T., M.T.
Suwito, S.T., M.T.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industry Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

**PERANCANGAN SISTEM STABILISASI KAMERA
TIGA SUMBU DENGAN METODE KONTROL
FUZZY UNTUK *MOBILE SURVEILLANCE ROBOT***

TUGAS AKHIR

**Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik**

Pada

Bidang Studi Elektronika

Jurusan Teknik Elektro

Institut Teknologi Sepuluh Nopember

Menyetujui :

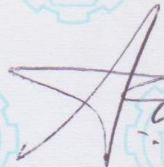
Dosen Pembimbing I,

Dosen Pembimbing II,



Dr. Muhammad Rivai, S.T., M.T.

NIP. 196904261994031003



Suwito, S.T., M.T.

NIP. 198101052005011004



ABSTRAK

Perancangan Sistem Stabilisasi Kamera Tiga Sumbu dengan Metode Kontrol Fuzzy untuk *Mobile Surveillance Robot*

Fahrezi Alwi Muhammad
2212100150

Dosen Pembimbing I : Dr. Muhammad Rivai, ST., MT.
Dosen Pembimbing II : Suwito, ST., MT.

Abstrak

Robot pengintaian (*surveillance robot*) sering digunakan pada militer untuk melakukan tugas pengintaian sehingga tidak perlu membahayakan nyawa manusia karena dikendalikan dari jarak jauh. Robot pengintai memiliki kamera yang diletakkan di atas robot. Ketika melakukan pengamatan menggunakan kamera ini, seringkali pengamatan terganggu akibat guncangan-guncangan yang terjadi pada kamera. Guncangan ini disebabkan karena permukaan jalan yang dilalui oleh robot tidak rata.

Sistem stabilisasi kamera adalah perangkat yang digunakan untuk menghilangkan guncangan dan menjaga posisi kamera agar kamera dapat mengambil gambar dengan baik pada suatu sudut pandang tertentu. Pada tugas akhir ini, dibuat sebuah sistem stabilisasi untuk robot pengintai dengan dua buah sensor *gyroscope* MPU-6050 untuk mengetahui kecepatan sudut guncangan dan kecepatan sudut kamera, mikrokontroler Arduino Mega sebagai pusat kontrol dan tiga buah motor DC *brushless* sebagai aktuator. Metode kontrol yang ditanamkan pada sistem terdiri dari tiga sistem Fuzzy untuk menangani sumbu *pitch*, *roll*, dan *yaw*.

Pada tugas akhir ini diujikan dua metode. Metode pertama menggunakan satu sensor *gyroscope* sebagai nilai *feedback*, metode kedua menggunakan dua sensor *gyroscope* sebagai nilai *feedback* dan *set point*. Standar deviasi pada pengujian perekaman video tanpa kontrol untuk *pixel* x 40.57 dan *pixel* y 32.95. Standar deviasi dengan metode pertama untuk *pixel* x 24.73 adalah dan *pixel* y 21.73. Sedangkan standar deviasi metode kedua untuk *pixel* x 16.70 dan *pixel* y 22.44.

Kata kunci : Fuzzy, gimbal, motor *brushless* DC, sensor *gyroscope*

ABSTRACT

Design of Three Axis Camera Stabilizer with Fuzzy Control Method for Mobile Surveillance Robot

Fahrezi Alwi Muhammad
2212100150

Advisor 1 : Dr. Muhammad Rivai, ST., MT.
Advisor 2 : Suwito, ST., MT.

Abstract

Surveillance robot is often used in the military to perform surveillance tasks that do not need to endanger human life because it is controlled remotely. Robot has surveillance cameras have been placed at the top of the robot. When making observations using this camera, observation is often disturbed due to shocks occur on the camera. These shocks caused by the road surface through which the robot is not flat.

Camera stabilization system is a device used to eliminate the shock and keep the camera so that the camera can take pictures with well at a certain angle. In this final project, will be designed a stabilization system for surveillance robot with two MPU-6050 gyroscope sensor to determine the angular velocity and angular velocity shocks camera, Arduino Mega as a controller and three brushless DC motors as actuators. The control method is embedded in a system consisting of three Fuzzy system to handle pitch, roll, and yaw axis.

In this final project tested two methods. The first method uses a gyroscope sensor as feedback value, the second method uses two gyroscope sensors as feedback and set point value. Standard deviation in test recording video without controls for pixel and pixel x 40.57 y 32.95. The standard deviation of the first method for pixel x and pixel y 24.73 is 21.73, while the standard deviation of the second method for pixel x and pixel y 22:44 16.70.

Keywords: Fuzzy, gimbal, *gyroscope* sensor, motor *brushless* DC

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR TABEL	xi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	1
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi Penelitian	2
1.6 Sistematika Penulisan	3
1.7 Relevansi	4
BAB II DASAR TEORI	5
2.1 Orientasi <i>Roll, Pitch, Yaw</i>	5
2.2 Gimbal	5
2.3 Sensor <i>Gyroscope</i>	6
2.4 Motor <i>Brushless</i> DC	8
2.4.1 Konstruksi Motor <i>Brushless</i> DC	8
2.4.2 Konfigurasi Motor <i>Brushless</i> DC	9
2.4.3 Belitan Stator Motor <i>Brushless</i> DC	9
2.4.4 Prinsip Kerja Motor <i>Brushless</i> DC	9
2.5 Mikrokontroler	12
2.6 Metode Kontrol Fuzzy	12
2.6.1 Fuzzifikasi	13
2.6.2 Pembuatan <i>rule</i>	14
2.6.3 Defuzzifikasi	15
2.7 Kamera	16
BAB III PERANCANGAN SISTEM	19
3.1 Perancangan Perangkat Keras	19
3.1.1 Gambaran Sistem	19
3.1.2 Hubungan Kabel dalam Sistem	23
3.2 Perancangan Perangkat Lunak	24
3.2.1 Diagram Blok Sistem Metode Pertama	24
3.2.2 Diagram Blok Sistem Metode Kedua	25
3.2.3 Register <i>Clock</i>	26
3.2.4 Register <i>Gyroscope</i>	27

3.2.5	Diagram Alir Pengaturan Register Sensor	28
3.2.6	Pembacaan Data <i>Gyroscope</i>	29
3.2.7	Diagram Alir Pembacaan Data <i>Gyroscope</i>	30
3.2.8	Perancangan Kontrol Fuzzy	30
3.2.9	Diagram Alir Program Utama Mikrokontroler.....	33
3.2.10	Perancangan Program untuk Pengujian Lapangan Menggunakan OpenCV	34
3.2.11	Diagram Alir Program untuk Pengujian Lapangan Menggunakan OpenCV	35
BAB IV PENGUKURAN DAN ANALISIS SISTEM.....		37
4.1	Pengujian Sensor <i>Gyroscope</i>	37
4.2	Pengujian Kestabilan Sistem dengan Guncangan Terukur ...	40
4.3	Pengujian Kestabilan Sistem pada Lapangan	44
BAB V PENUTUP		47
5.1	Kesimpulan.....	47
5.2	Saran.....	47
DAFTAR PUSTAKA.....		49
LAMPIRAN		51
	Potongan Program pada Arduino Mega	51
	Program untuk Pengujian Menggunakan OpenCV	58
BIOGRAFI PENULIS		61

DAFTAR TABEL

Tabel 3.1	Pengaturan Register Sumber <i>Clock</i>	26
Tabel 3.2	Register Konfigurasi <i>Gyroscope</i>	27
Tabel 3.3	Pengaturan Skala Penuh dan Skala Pemfaktor MPU-6050	27
Tabel 3.4	Register Data <i>Gyroscope</i> pada MPU-6050.....	29
Tabel 3.5	<i>Rule Base</i> Sistem Kontrol Fuzzy	31
Tabel 4.1	Hasil Pengujian Guncangan Terukur	43
Tabel 4.2	Hasil Pengujian Lapangan	46

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1	Orientasi Sumbu Roll, Pitch, Yaw	5
Gambar 2.2	Gimbal untuk Kapal Laut	6
Gambar 2.3	Efek Coriolis Benda yang Bergerak Lurus pada Cakram Berputar	6
Gambar 2.4	Modul MEMS Gyroscope MPU-6050	7
Gambar 2.5	Representasi Sinyal Input DC Sinusoidal pada Sinyal PWM	10
Gambar 2.6	Prinsip Kerja Motor DC Brushless dalam 6 Fase	11
Gambar 2.7	Arduino Mega	12
Gambar 2.8	Proses Kontrol Logika Fuzzy	13
Gambar 2.9	Proses Penalaran Max-Min	15
Gambar 2.10	Perbedaan Cara Kerja Sensor CCD dan CMOS	17
Gambar 3.1	Tampak Samping Sistem.....	19
Gambar 3.2	Tampak Depan Sistem	19
Gambar 3.3	Arduino Mega dan <i>Driver Shield</i>	20
Gambar 3.4	Sensor <i>Gyroscope</i> Pertama.....	21
Gambar 3.5	Sensor <i>Gyroscope</i> Kedua	21
Gambar 3.6	Pegas yang Masih Terpasang	22
Gambar 3.7	Pegas yang Telah Dilepas	22
Gambar 3.8	Keseluruhan Sistem.....	23
Gambar 3.9	Hubungan Kabel Arduino Mega Dengan Sensor, Motor <i>Driver</i> dan Motor <i>Brushless</i> DC.....	23
Gambar 3.10	Diagram Blok Sistem Metode Pertama	24
Gambar 3.11	Diagram Blok Sistem Metode Kedua.....	25
Gambar 3.12	Diagram Alir Pengaturan Register Sensor	28
Gambar 3.13	Diagram Alir Pembacaan Data <i>Gyroscope</i>	30
Gambar 3.14	<i>Membership Function Input Error</i>	31
Gambar 3.15	<i>Membership Function Input Delta Error</i>	31
Gambar 3.16	Posisi Nilai x Anggota <i>Output Singleton</i>	32
Gambar 3.17	Diagram Alir Program Utama Mikrokontroler.....	33
Gambar 3.18	Diagram Alir Program Pengujian Lapangan	35
Gambar 4.1	Offset Sumbu X Sensor <i>Gyroscope</i> Pertama.....	37
Gambar 4.2	Offset Sumbu Y Sensor <i>Gyroscope</i> Pertama.....	38
Gambar 4.3	Offset Sumbu Z Sensor <i>Gyroscope</i> Pertama	38

Gambar 4.4	Offset Sumbu X Sensor <i>Gyroscope</i> Kedua.....	39
Gambar 4.5	Offset Sumbu Y Sensor <i>Gyroscope</i> Kedua.....	39
Gambar 4.6	Offset Sumbu Z Sensor <i>Gyroscope</i> Kedua	40
Gambar 4.7	Respon Sistem Sumbu X dengan Metode Pertama....	40
Gambar 4.8	Respon Sistem Sumbu X dengan Metode Kedua	41
Gambar 4.9	Respon Sistem Sumbu Y dengan Metode Pertama....	41
Gambar 4.10	Respon Sistem Sumbu Y dengan Metode Kedua	42
Gambar 4.11	Respon Sistem Sumbu Z dengan Metode Pertama	42
Gambar 4.12	Respon Sistem Sumbu Z dengan Metode Kedua	43
Gambar 4.13	Pengolahan Citra Objek yang Diamati	44
Gambar 4.14	Posisi Perubahan <i>Pixel</i> Objek yang Diamati Tanpa Menggunakan Kontrol	45
Gambar 4.15	Posisi Perubahan <i>Pixel</i> Objek yang Diamati Dengan Menggunakan Metode Pertama	45
Gambar 4.16	Posisi Perubahan <i>Pixel</i> Objek yang Diamati dengan Menggunakan Metode Kedua	46

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi robot yang cepat saat ini menjadikan kualitas kehidupan manusia semakin baik. Robot adalah sebuah mesin yang dapat bekerja secara terus-menerus baik secara otomatis maupun terkendali [1]. Robot digunakan untuk membantu manusia dalam mengerjakan hal-hal yang sulit sehingga dapat menghemat waktu atau yang tidak bisa dilakukan manusia secara langsung. Sebagai contoh, robot digunakan dalam industri perakitan kendaraan, industri makanan, industri tekstil, hingga industri militer. Salah satu robot yang saat ini sering digunakan oleh militer adalah robot pengintai (*surveillance robot*). Dengan robot ini, para prajurit tidak perlu mengintai suatu lokasi yang berbahaya secara langsung karena robot ini dikendalikan dari jarak jauh. Robot ini digunakan untuk melakukan tugas pengintaian sehingga tidak perlu membahayakan nyawa manusia. Robot pengintai memiliki kamera yang diletakkan di atas robot. Kamera tersebut akan mentransmisikan video secara langsung ke pusat pengendali sehingga pengguna dapat melakukan pengamatan di suatu tempat.

Ketika melakukan pengamatan menggunakan kamera jarak jauh ini, seringkali pengamatan terganggu akibat guncangan-guncangan yang terjadi pada kamera. Guncangan ini disebabkan karena permukaan jalan yang dilalui oleh robot tidak rata. Pada penelitian tugas akhir ini muncul suatu ide untuk membuat sistem stabilisasi kamera tiga sumbu yang menggunakan Fuzzy sebagai metode kontrolnya. Selama ini, metode pengontrolan pada aplikasi sistem stabilisasi kamera menggunakan kontrol PID. Penelitian ini bertujuan untuk mengetahui apakah metode Fuzzy cocok untuk diimplementasikan dalam sistem stabilisasi kamera. Sistem ini membaca sensor *gyroscope* untuk mengetahui kecepatan sudut guncangan dan kecepatan sudut kamera, dan tiga buah motor DC *brushless* sebagai aktuator.

1.2 Perumusan Masalah

Bedasarkan latar belakang di atas, dapat dirumuskan beberapa masalah, antara lain :

1. Mengidentifikasi guncangan pada sistem.
2. Implementasi kontrol Fuzzy untuk mengendalikan aktuator.

1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Sistem mampu mengidentifikasi guncangan menggunakan sensor *gyroscope*.
2. Kontrol Fuzzy yang dibuat dapat mengendalikan motor DC *brushless*.

1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Robot yang dipasang kamera adalah mobil mainan RC.
2. Pergerakan robot diujikan pada bidang yang tidak rata dengan pergerakan yang lurus.
3. Hasil perekaman video hanya untuk pengujian, tidak diproses lebih lanjut untuk aplikasi *tracking*.

1.5 Metodologi Penelitian

Tugas akhir ini menggunakan metode penelitian sebagai berikut:

1. Studi Literatur

Tahapan ini adalah segenap kegiatan pencarian dan pengkajian referensi yang berhubungan dengan topik tugas akhir. Referensi studi literatur didapatkan melalui buku, jurnal ilmiah, dan browsing melalui internet yang berhubungan dengan judul tugas akhir ini. Studi literatur dilakukan secara beriringan dengan penelitian.

2. Perancangan Sistem

Perancangan perangkat keras meliputi perancangan sistem yang terdiri dari Perancangan ini dilakukan dengan studi literatur sebelumnya. Perancangan ini akan meliputi perancangan *hardware* dan *software*. Perancangan *hardware* meliputi penempatan sensor *gyroscope*, dan pembuatan driver shield. Sedangkan frame gimbal yang sudah terpasang motor DC *brushless* memakai yang sudah tersedia di pasaran. Perancangan *software* akan dilakukan pada Arduino Mega. Segenap kontrol seperti kontrol motor menggunakan metode

Fuzzy dilakukan pada tahapan ini melalui perancangan *software*.

3. Pengujian dan Analisa Data

Pada tahapan ini akan dilakukan akan dilakukan berbagai pengujian terhadap sistem untuk melihat respon sistem. *Sensor drifting* adalah salah satu akan diuji. Untuk kontrol akan dilakukan pengujian untuk mengetahui respon motor dari sinyal masukan oleh sensor. Lalu pengujian keseluruhan sistem akan dilakukan dengan merekam suatu objek yang di-*tracking* pergerakannya.

4. Penarikan Kesimpulan dan Saran

Kesimpulan ditarik dari pengujian, analisa data, dan juga referensi. Pada kesimpulan akan ditarik garis besar jawaban dari permasalahan yang dianalisis. Selain itu, juga akan diberikan saran sebagai masukan berkaitan dengan apa yang telah dilakukan. Kesimpulan dan saran dapat digunakan untuk pengembangan penelitian selanjutnya.

5. Penyusunan Buku Tugas Akhir

Tahap penyusunan buku tugas akhir akan dilakukan beriringan dengan penelitian. Penyusunan buku tugas akhir adalah bentuk laporan tertulis dari penelitian. Penulisan buku tugas akhir dilakukan dengan sistematika penulisan yang telah ditentukan.

1.6 Sistematika Penulisan

Laporan tugas akhir ini ditulis berdasarkan format yang telah ditentukan oleh Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember. Tugas akhir ini terdiri dari lima bab yang akan dijelaskan pada poin di bawah.

- Bab 1 : Pendahuluan
Bab ini menjelaskan latar belakang, perumusan masalah, tujuan, sistematika penulisan, metodologi, dan relevansi tugas akhir.
- Bab 2 : Dasar Teori
Bab ini menjelaskan tentang berbagai macam teori pendukung dalam penulisan tugas akhir. Bab ini meliputi dasar teori dari orientasi sistem, gimbal, sensor *gyroscope*, motor DC *brushless*, metode kontrol Fuzzy, dan kamera.

- Bab 3 : Perancangan Sistem
- Bab 4 : Pengujian dan Analisis
- Bab 5 : Penutup

1.7 Relevansi

Mata kuliah pendukung tugas akhir ini adalah Sistem Mikroprosesor dan Mikrokontroler, Sensor dan Aktuator, Dasar Sistem Elektronika Cerdas, Sistem Kontrol Elektronika, dan Penginderaan Visual.

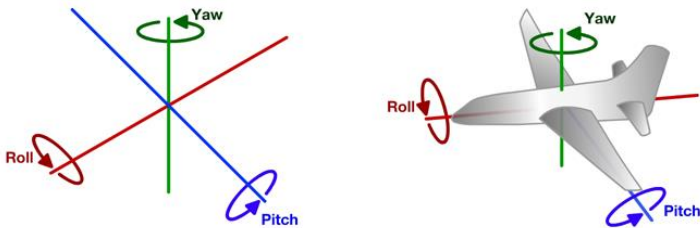
Hasil dari tugas akhir ini diharapkan dapat memberikan sumbangsih penelitian dalam bidang elektronika dan juga bidang penelitian untuk aplikasi stabilisasi kamera.

BAB II

DASAR TEORI

Pada bab ini akan dijelaskan berbagai macam dasar teori yang digunakan dalam perancangan sistem.

2.1 Orientasi *Roll, Pitch, Yaw*



Gambar 2.1 Orientasi Sumbu Roll, Pitch, Yaw [2]

Dalam dunia penerbangan, dikenal orientasi *roll*, *pitch*, *yaw* dalam ranah tiga dimensi yang tersusun atas tiga buah sumbu (X, Y, Z) dan sebuah titik origin (*center of gravity*). Orientasi *roll* adalah orientasi dimana sumbu putar berada pada sumbu X atau sumbu longitudinal. Orientasi *pitch* memiliki sumbu putar pada sumbu Y atau sumbu lateral. Lalu orientasi *yaw* memiliki sumbu putar pada sumbu Z atau sumbu vertikal. Ilustrasi *roll*, *pitch*, *yaw* ditunjukkan pada Gambar 2.1

2.2 Gimbal

Gimbal merupakan platform yang dapat berputar yang paling tidak terdiri dari satu buah cincin. Pada cincin tersebut memiliki sepasang *bearing* untuk sumbu poros putar [3]. Gimbal banyak difungsikan untuk keperluan pesawat terbang dan kapal laut. Kegunaan dari gimbal untuk kompas kapal adalah mempertahankan posisi kompas agar tetap pada sumbu horizontal agar informasi arah yang didapatkan benar karena pada kapal selalu terdapat guncangan yang berasal dari gelombang laut.

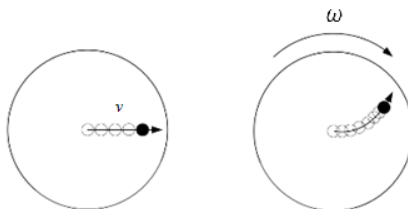


Gambar 2.2 Gimbal untuk Kapal Laut [4]

2.3 Sensor *Gyroscope*

Gyroscope adalah sensor yang mendeteksi kecepatan sudut eksternal. *Gyroscope* merupakan salah satu sensor Inertial Measurement Unit (IMU). IMU merupakan sistem untuk mengukur pergerakan suatu objek pada suatu ruang yang relatif terhadap frame inersia [5]. Satuan yang dipergunakan pada *gyroscope* adalah derajat/detik ($^{\circ}/s$). Sensor *gyroscope* juga dapat difungsikan untuk sensor penentuan orientasi. Kecepatan sudut merupakan turunan pertama dari orientasi, sehingga untuk mendapatkan nilai orientasi dari *gyroscope* perlu dilakukan integrasi data.

Gyroscope memiliki kesamaan prinsip kerja dengan *accelerometer*. Namun, *gyroscope* dapat mengubah input kecepatan sudut perpindahan massanya. Di dalam ilmu fisika, jika sebuah benda bergerak lurus dalam kerangka yang berputar akan terlihat berbelok oleh pengamat yang diam pada kerangka tersebut. Fenomena tersebut dikenal dengan istilah efek coriolis. Efek coriolis dapat dijelaskan melalui Gambar 2.3.



Gambar 2.3 Efek Coriolis Benda yang Bergerak Lurus pada Cakram Berputar

Dari Gambar 2.3 dapat dilihat bahwa ketika sebuah partikel bergerak lurus (pada *gyroscope*, partikel ini bergetar lurus) dengan kecepatan v pada cakram yang diputar searah jarum jam, maka lintasan partikel akan mengalami pembelokan. Hal ini disebabkan karena adanya pengaruh rotasi piring terhadap gerak dari peluru. Semakin cepat cakram berputar, semakin besar pula pembelokan partikel yang terjadi. Efek coriolis ini akan menghasilkan gaya rotasi yang arahnya tegak lurus dengan kecepatan v yang besarnya sesuai dengan persamaan:

$$\vec{F}_c = m\vec{a}_c = -2m\vec{\omega} \times \vec{v} \quad (2.1)$$

Di mana:

F_c = Gaya Coriolis (N)

m = massa (kg)

a_c = percepatan coriolis (m/s²)

ω = Kecepatan sudut (rad/s)

Dari Persamaan 2.1 didapat bahwa besarnya percepatan coriolis a_c berbanding lurus secara proporsional dengan nilai ω . Selanjutnya percepatan coriolis dikonversikan ke dalam besaran elektrik sehingga kecepatan putar dapat dengan mudah diukur dan diolah.

Dalam penelitian ini, *gyroscope* yang digunakan adalah MEMS (Microelectromechanical System) *gyroscope*. Dengan teknologi MEMS, sangat dimungkinkan untuk membuat *gyroscope* dalam ukuran yang sangat kecil, meskipun di dalamnya terdapat sistem mekanik yang rumit.



Gambar 2.4 Modul MEMS Gyroscope MPU-6050 [6]

Dalam penerapan MEMS *gyroscope* sebagai sensor untuk pengukuran kecepatan sudut terdapat masalah *drifting*. *Drifting* adalah keadaan di mana *gyroscope* diam namun menunjukkan adanya offset kecepatan sudut. Offset pada *gyroscope* dapat dihilangkan dengan mengurangi atau menambahkan nilai tersebut hingga hasil akhirnya nol pada program.

2.4 Motor *Brushless* DC

Motor *brushless* DC (BLDC) merupakan jenis motor DC yang tidak menggunakan sikat (*brush*) sebagai komutatornya seperti halnya pada motor *brushed* DC. Karena tidak menggunakan sikat maka komutasi dari motor *brushless* DC dilakukan secara elektrikal dengan bantuan rangkaian *switching* dan pemroses seperti mikrokontroler. Motor *brushless* DC banyak digunakan sebagai aktuator dalam sistem elektromekanik karena memiliki kepresisian tinggi. Contoh aplikasinya adalah lengan robot, *conveyor*, dan *positioning system* [7].

Terdapat keuntungan dan kekurangan dari motor *brushless* DC dibandingkan dengan motor *brushed* DC. Keuntungan dari motor *brushless* DC antara lain:

- Rentang kecepatan yang tinggi
- Biaya perawatan lebih rendah
- Efisiensi tinggi

Sedangkan untuk kekurangan dari motor *brushless* DC antara lain:

- Pengontrolan motor yang rumit
- Menggunakan rangkaian elektronika yang kompleks

Untuk mengenal motor *brushless* DC, berikut dijelaskan secara singkat mengenai konstruksi, konfigurasi, belitan kumparan stator, dan prinsip kerja dari motor *brushless* DC.

2.4.1 Konstruksi Motor *Brushless* DC

Secara umum motor *brushless* DC memiliki dua buah bagian utama yaitu stator dan rotor. Stator adalah bagian dari motor yang diam/statis. Pada motor *brushless* DC memiliki stator yang konstruksinya berbeda dengan motor DC *brushed*. Motor DC *brushed* memiliki stator yang difungsikan sebagai tempat meletakkan magnet

permanen. Sedangkan untuk motor *brushless* DC, stator difungsikan untuk tempat meletakkan belitan kumparan. Sedangkan rotor adalah bagian dari motor yang berputar. Rotor pada motor *brushless* DC tersusun atas magnet permanen. Banyaknya magnet permanen ada rotor akan mempengaruhi presisi gerakan motor. Semakin banyak kutub magnet maka semakin tinggi presisi motor.

2.4.2 Konfigurasi Motor *Brushless* DC

Terdapat dua jenis konfigurasi fisik pada motor *brushless* DC. Yaitu *outer-rotor brushless* DC dan *inner-rotor brushless* DC. *Outer-rotor brushless* DC menempatkan belitan kumparan yang dikelilingi oleh magnet permanen yang dikelilingi *casing*. Sedangkan *inner-rotor brushless* DC menempatkan magnet permanen didalam dan dikelilingi oleh belitan kumparan.

2.4.3 Belitan Stator Motor *Brushless* DC

Terdapat dua buah tipe belitan stator pada motor *brushless* DC, yaitu belitan bintang (Y) dan belitan delta (Δ). Motor *brushless* DC dengan belitan bintang memiliki kecepatan yang lebih pelan dibanding motor dengan belitan delta, sebaliknya motor belitan bintang memiliki torsi lebih besar dibanding motor belitan delta [8]. Hubungan belitan bintang dan delta disajikan pada Persamaan (2.2) dan Persamaan (2.3)

$$M_Y = \sqrt{3} M_{\Delta} \quad (2.2)$$

$$\omega_Y = \frac{1}{\sqrt{3}} \omega_{\Delta} \quad (2.3)$$

di mana M_Y adalah torsi pada belitan bintang M_{Δ} adalah torsi belitan delta, ω_Y merupakan kecepatan sudut pada belitan bintang, dan ω_{Δ} adalah kecepatan sudut pada belitan delta.

2.4.4 Prinsip Kerja Motor *Brushless* DC

Motor *brushless* DC dapat berputar apabila pada stator diberikan tegangan komutasi berupa sinyal sinusoidal, sinyal kotak atau sinyal trapezoidal tiga fasa dengan selisih 120° antar fasa yang berasal dari sinyal input DC yang telah dikonversi menjadi AC oleh *inverter* internal.

Arus yang melewati belitan kumparan stator akan menimbulkan medan magnet. Persamaan dari medan magnet tersebut adalah

$$B = \frac{\mu N I}{2l} \quad (2.4)$$

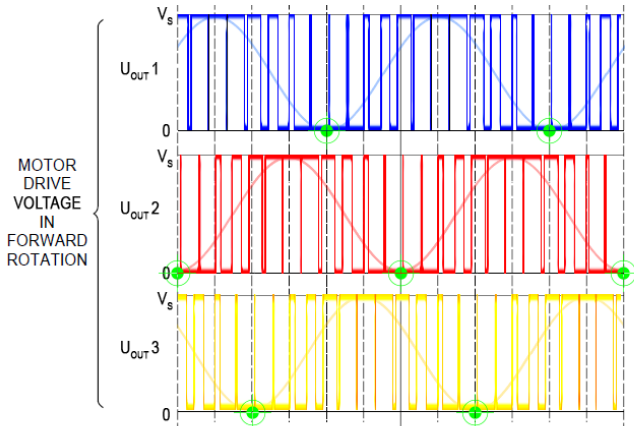
di mana B adalah medan magnet, N adalah jumlah lilitan, I adalah arus yang melewati belitan, l adalah panjang belitan dan μ adalah permeabilitas bahan.

Pemberian arus AC akan mengakibatkan medan magnet dan polarisasi belitan berubah-ubah terhadap waktu. Perubahan tersebut akan menimbulkan putaran motor *brushless* DC dengan kecepatan

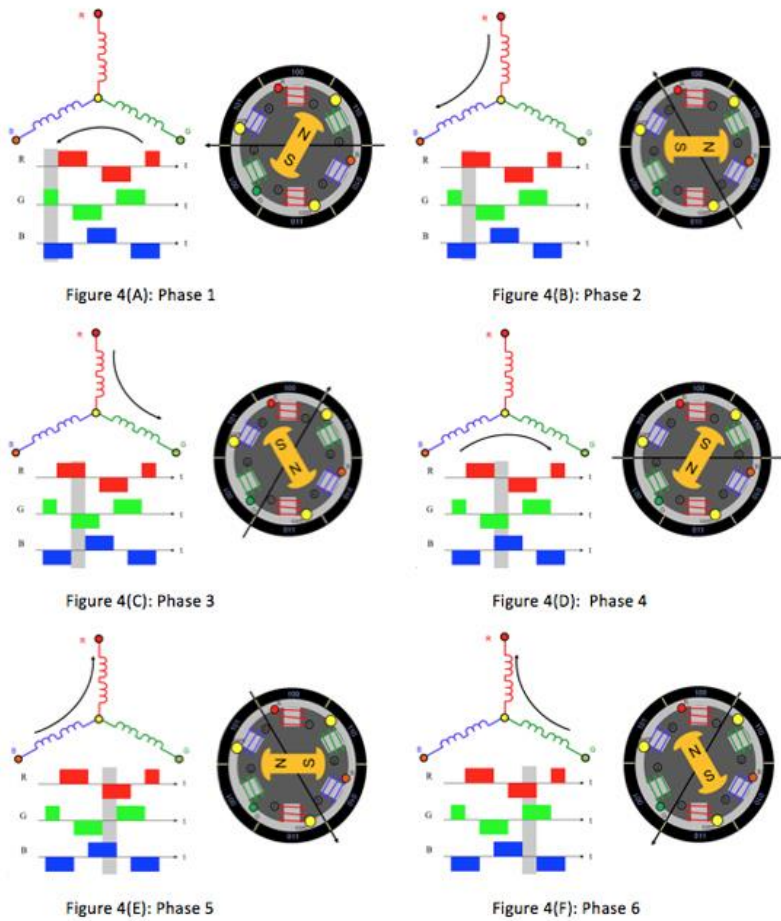
$$N_s = \frac{120f}{p} \quad (2.5)$$

di mana N_s merupakan kecepatan medan putar stator, f adalah frekuensi tegangan masukan motor dalam satuan Hz, dan p adalah jumlah kutub pada rotor.

Penelitian ini menggunakan sinyal sinusoidal DC yang telah direpresentasikan dengan modulasi lebar pulsa (PWM) sebagai sinyal kontrol motor *brushless* DC. PWM mengatur kecepatan motor maupun posisi sudut motor *brushless* DC menggunakan tegangan rata-rata output dan frekuensi tertentu yang telah disesuaikan.



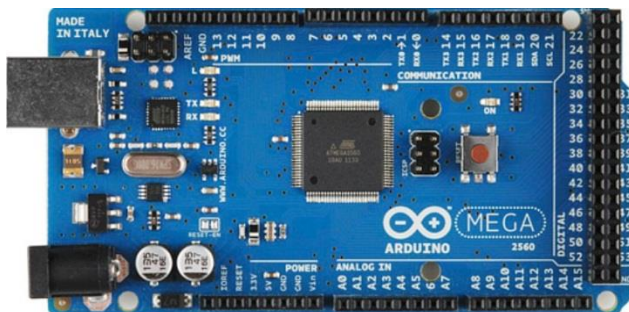
Gambar 2.5 Representasi Sinyal Input DC Sinusoidal pada Sinyal PWM [9]



Gambar 2.6 Prinsip Kerja Motor DC Brushless dalam 6 Fase [10]

2.5 Mikrokontroler

Mikrokontroler adalah sebuah chip yang berfungsi sebagai pengontrol rangkaian elektronik dan umunya dapat menyimpan program di dalamnya. Mikrokontroler adalah sebuah chip yang berfungsi sebagai pengontrol rangkaian elektronik dan umunya dapat menyimpan program yang umumnya terdiri dari CPU (Central Processing Unit), memori, I/O tertentu dan unit pendukung seperti Analog-to-Digital Converter (ADC) yang sudah terintegrasi di dalamnya. Kelebihan utama dari mikrokontroler ialah tersedianya RAM dan peralatan I/O pendukung sehingga ukuran board mikrokontroler menjadi sangat ringkas.



Gambar 2.7 Arduino Mega [11]

2.6 Metode Kontrol Fuzzy

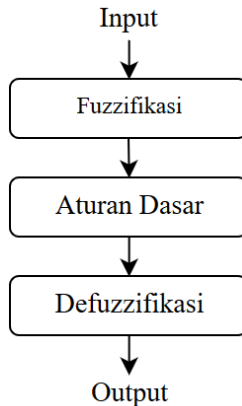
Logika fuzzy adalah suatu cara yang tepat untuk memetakan suatu ruang input dalam suatu ruang output dan memiliki nilai yang berlanjut. Kelebihan logika fuzzy ada pada kemampuan penalaran secara bahasa. Sehingga, dalam perancangannya tidak memerlukan persamaan matematis yang kompleks dari objek yang akan dikendalikan.

Sistem kontrol logika fuzzy disebut juga sistem Inferensi Fuzzy (Fuzzy Inference System/FIS) atau fuzzy inference engine adalah sistem yang dapat melakukan penalaran dengan prinsip serupa seperti manusia melakukan penalaran dengan nalurinya.

Terdapat beberapa jenis FIS yang dikenal yaitu Mamdani, Sugeno dan Tsukamoto. FIS yang paling mudah

dimengerti, karena paling sesuai dengan naluri manusia adalah FIS Mamdani. FIS tersebut bekerja berdasarkan kaidah-kaidah linguistik dan memiliki algoritma fuzzy yang menyediakan sebuah aproksimasi untuk dimasuki analisa matematik [12].

Sistem kendali logika fuzzy terdiri dari beberapa tahapan seperti pada diagram berikut.



Gambar 2.8 Proses Kontrol Logika Fuzzy

Proses dalam kontrol logika *fuzzy* ditunjukkan pada gambar di atas. Input yang diberikan kepada adalah berupa bilangan tertentu dan output yang dihasilkan juga harus berupa bilangan tertentu. Aturan-aturan dalam bahasa linguistik dapat digunakan sebagai input yang bersifat teliti harus dikonversikan terlebih dahulu, lalu melakukan penalaran berdasarkan aturan-aturan dan mengkonversi hasil penalaran tersebut menjadi output yang bersifat teliti [12].

2.6.1 Fuzzifikasi

Fuzzifikasi adalah pemetaan nilai input yang merupakan nilai tegas ke dalam fungsi keanggotaan himpunan fuzzy, untuk kemudian diolah di dalam mesin penalaran.

$$\text{fuzzifikasi : } x \rightarrow \mu(x) \quad (2.6)$$

2.6.2 Pembuatan *rule*

Aturan dasar dalam kendali logika fuzzy adalah aturan implikasi dalam bentuk “jika ... maka ...”. Aturan dasar tersebut ditentukan dengan bantuan seorang pakar yang mengetahui karakteristik objek yang akan dikendalikan.

Contoh bentuk implikasi yang digunakan adalah sebagai berikut.

Jika $X = A$ dan $Y = B$ maka $Z = C$.

Pada tahapan ini sistem menalar nilai masukan untuk menentukan nilai keluaran sebagai bentuk pengambil keputusan. Sistem terdiri dari beberapa aturan, maka kesimpulan diperoleh dari kumpulan dan korelasi antar aturan.

Ada 3 metode yang digunakan dalam melakukan inferensi sistem fuzzy, yaitu max, additive dan probabilistik OR.

Pada metode max, solusi himpunan fuzzy diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah fuzzy, dan mengaplikasikannya ke output dengan menggunakan operator OR (union). Secara umum dapat ditulis [12]:

$$\mu_{df}(x_i) \leftarrow \max(\mu_{df}(x_i), \mu_{kf}(x_i)) \quad (2.7)$$

Selain itu, salah satu model penalaran yang banyak digunakan adalah max-min. Dalam penalaran ini, pertama-tama dilakukan proses operasi min sinyal keluaran lapisan fuzzyfikasi, kemudian diteruskan dengan operasi max untuk mencari nilai keluaran yang selanjutnya akan didefuzzyfikasikan sebagai bentuk keluaran pengendali [13]. Operasi max-min tersebut dapat dinyatakan sebagai berikut.

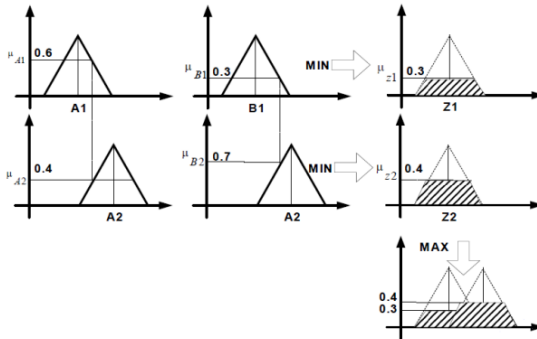
Operasi min atau irisan

$$\begin{aligned} a \wedge b &= \min(a, b) = a \text{ if } a \leq b \\ &= b \text{ if } a > b \end{aligned} \quad (2.8)$$

Operasi max atau gabungan

$$\begin{aligned}
 a \vee b = \max(a,b) &= a \text{ if } a \geq b \\
 &= b \text{ if } a < b
 \end{aligned}
 \tag{2.9}$$

Proses penalaran max-min dijelaskan dalam grafik berikut.



Gambar 2.9 Proses Penalaran Max-Min [13]

2.6.3 Defuzzifikasi

Defuzzifikasi merupakan kebalikan dari fuzzifikasi, yaitu pemetaan dari himpunan fuzzy ke himpunan tegas. Input dari proses defuzzifikasi adalah suatu himpunan fuzzy yang diperoleh dari komposisi aturan- aturan fuzzy. Hasil dari defuzzifikasi ini merupakan output dari sistem kendali logika fuzzy.

Defuzzifikasi dideskripsikan sebagai:

$$Z^* = \text{defuzzifier}(Z) \tag{2.10}$$

dengan

Z = hasil penalaran fuzzy

Z^* = keluaran kendali logika fuzzy

defuzzifier = fungsi defuzzifikasi [13]

Metode defuzzifikasi antara lain [13]:

1. Metode Maximum

Metode ini juga dikenal dengan metode puncak, yang nilai keluarannya dibatasi oleh fungsi $\mu_c(z^*) > \mu_c(z)$.

2. Metode titik tengah

Metode titik tengah juga disebut metode pusat area. Metode ini lazim dipakai dalam proses defuzzifikasi. Keluaran dari metode ini adalah titik tengah dari hasil proses penalaran.

3. Metode rata-rata

Metode ini digunakan untuk fungsi keanggotaan keluaran yang simetris. Keluaran dari metode ini adalah nilai rata-rata dari hasil proses penalaran.

4. Metode penjumlahan titik tengah

Keluaran dari metode ini adalah penjumlahan titik tengah dari hasil proses penalaran.

5. Metode titik tengah area terbesar

Dalam metode ini, keluarannya adalah titik pusat dari area terbesar yang ada.

2.7 Kamera

Kamera digital merupakan jenis kamera yang proses pengambilan gambarnya dilakukan secara digital, dengan media perekam/penyimpanan berupa memori (*flash*). Untuk beberapa jenis kamera digital ada pula yang dapat digunakan untuk merekam suara. Pada kamera digital ini, penggunaan elemen kimia telah digantikan dengan elemen sensor.

Sensor inilah yang akan mengatur kepekaan pencahayaannya menjadi "film digital" pada kamera-kamera modern yang beredar saat ini. Sampai saat ini jenis sensor pada kamera terbagi menjadi 2 jenis, yaitu CCD dan CMOS.

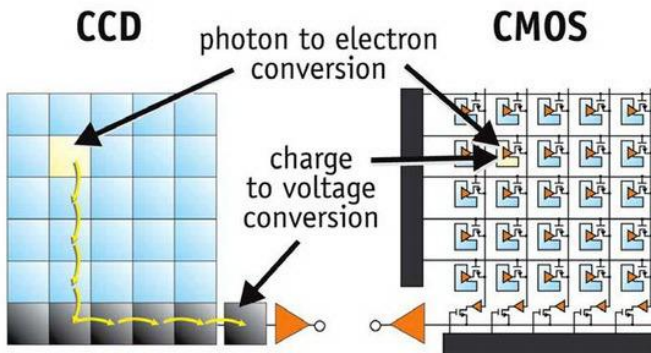
Baik sensor CCD (*Charge-Coupled Device*) maupun CMOS (*Complimentary Metal-Oxide Semiconductor*) berfungsi sama yaitu mengubah cahaya menjadi elektron. Meskipun kedua sensor tersebut

mempunyai fungsi yang sama yaitu untuk mengonversi cahaya menjadi elektron-elektron sehingga menjadi gambar-gambar digital, namun kedua memiliki perbedaan yang jauh.

Sensor yang digunakan di kamera digital seperti memiliki jutaan sel surya yang kecil dalam bentuk matrik dua dimensi. Masing-masing sel akan *men-transform* cahaya dari sebagian kecil gambar yang ditangkap menjadi elektron. Kedua sensor tersebut melakukan pekerjaan tersebut dengan berbagai macam teknologi yang ada.

Langkah selanjutnya adalah membaca nilai dari setiap sel di dalam gambar. Dalam kamera bersensor CCD, nilai tersebut dikirimkan ke dalam sebuah chip dan sebuah konverter analog ke digital mengubah setiap nilai *pixel* menjadi nilai digital. Dalam kamera bersensor CMOS, ada beberapa transistor dalam setiap pixel yang memperkuat dan memindahkan elektron dengan menggunakan kabel. Sensor CMOS lebih fleksibel karena membaca setiap pixel secara individual.

Sensor CCD memerlukan proses pembuatan secara khusus untuk menciptakan kemampuan memindahkan elektron ke chip tanpa distorsi. Dalam kata lain, sensor CCD menjadi lebih baik kualitasnya dalam ketajaman dan sensitivitas cahaya. Lain halnya, chip sensor CMOS dibuat dengan cara yang lebih tradisional dengan cara yang sama untuk membuat mikroprosesor.



Gambar 2.10 Perbedaan Cara Kerja Sensor CCD dan CMOS [14]

Karena proses pembuatannya tersebut, ada beberapa perbedaan mendasar dari sensor CCD dan CMOS:

CMOS (*Complementary Metal Oxide Semiconductor*).

- Tingkat kepekaan lebih rendah, karena terdapat beberapa transistor yang saling berdekatan pada setiap *pixel*.
- Sensor memiliki kemungkinan lebih besar untuk noise
- Umumnya menggunakan baterai atau sumber daya listrik yang lebih kecil/sedikit

CCD (*Charge Couple Device*).

- Menghasilkan gambar yang berkualitas tinggi
- Noise yang rendah (*low-noise*)
- Menggunakan listrik yang lebih besar, kurang lebih seratus kali lebih besar daripada sensor CMOS.

Dari sisi produksi sensor CCD telah diproduksi massal dalam jangka waktu yang lama, sehingga lebih matang dalam menjaga dan mengembangkan kualitasnya. Pabrikan sensor tersebut juga telah lebih memperbanyak serta merapatkan jumlah *pixel* yang mampu di tangkapnya. berbeda dengan chip CMOS yang diproduksi secara mikroprosesor yang umum, sehingga lebih murah jika dibandingkan dengan sensor CCD.

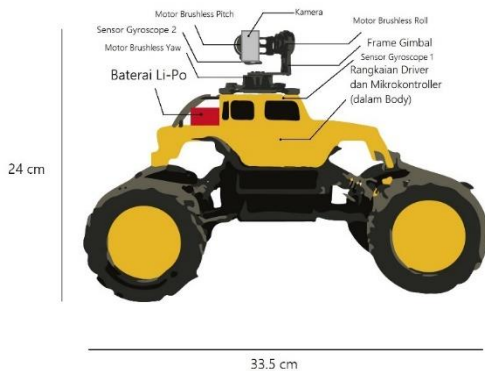
BAB III

PERANCANGAN SISTEM

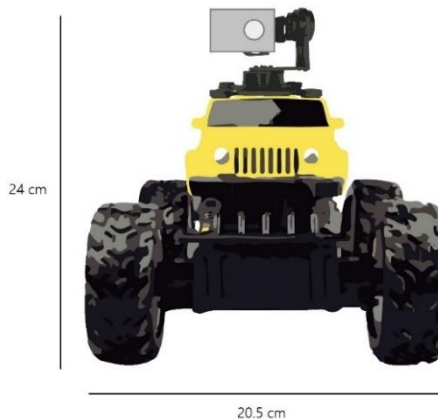
3.1 Perancangan Perangkat Keras

Pada bagian ini akan dijelaskan mengenai bagian-bagian perangkat keras dan penyusunannya.

3.1.1 Gambaran Sistem



Gambar 3.1 Tampak Samping Sistem

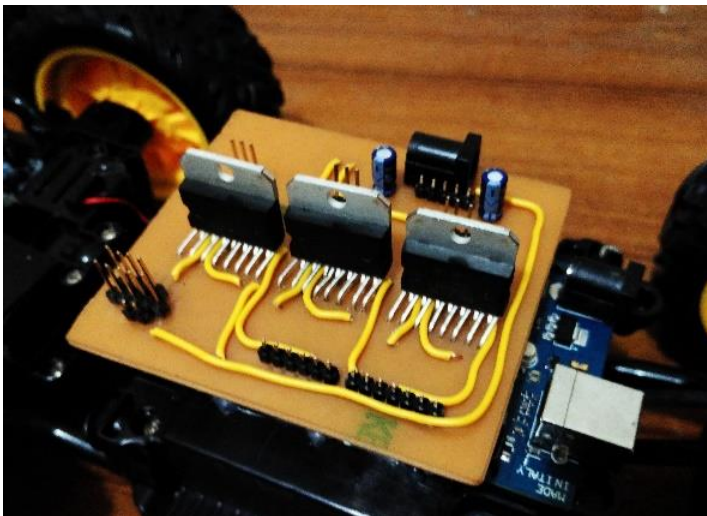


Gambar 3.2 Tampak Depan Sistem

Mobil RC yang digunakan dalam sistem untuk dijadikan mobil robot adalah Rastar Off-roader Rock Crawler 1:18. Pada sistem ini, *frame* gimbal dan baterai LiPo Tiger Power 2200mAh 2 cell 7.4V diletakkan di atas *body* mobil RC seperti pada Gambar 3.1. Pada penelitian ini tidak menggunakan baterai yang tersedia di bawah *chassis* untuk sumber daya mobil RC dikarenakan dapat mengganggu kestabilan sistem ketika mobil dijalankan.

Frame gimbal yang digunakan adalah suatu *frame* yang memiliki 3 derajat kebebasan (3 DOF). Dalam bidang ortogonal, *frame* ini memiliki sumbu pergerakan *roll*, *pitch* dan *yaw*. *Frame* gimbal yang digunakan adalah *frame* yang telah jadi dan terdapat di pasaran. *Frame* ini dijual tanpa merk. Motor *brushless* DC yang digunakan sebagai aktuator adalah motor *brushless* gimbal GBM2804. Motor ini sudah termasuk dalam penjualan *frame* gimbal. Motor ini memiliki tipe belitan bintang. Motor ini memiliki spesifikasi sebagai jumlah lilitan per *pole* 160 lilitan dan memiliki resistansi kumparan 2 jalur 12.5 ohm. Motor ini sudah termasuk dalam penjualan *frame*.

Arduino Mega dan *driver shield* diletakkan di atas *chassis* seperti ditunjukkan pada Gambar 3.3.



Gambar 3.3 Arduino Mega dan *Driver Shield*

Sensor *gyroscope* pertama yang berfungsi sebagai pemberi nilai set poin ditempel di bawah *body* mobil bagian atas sedangkan sensor *gyroscope* kedua yang berfungsi memberikan nilai *feedback* diletakkan di bawahudukan kamera pada *frame* seperti ditunjukkan pada Gambar 3.4 dan 3.5



Gambar 3.5 Sensor *Gyroscope* Pertama

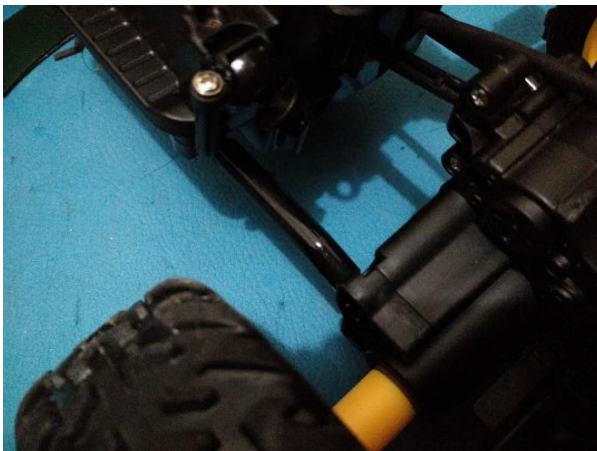


Gambar 3.4 Sensor *Gyroscope* Kedua

Pegas pada mobil RC yang berfungsi sebagai *damper* pada roda dilepas agar tidak mempengaruhi pengukuran seperti ditunjukkan pada Gambar 3.6 dan 3.7



Gambar 3.6 Pegas yang Masih Terpasang



Gambar 3.7 Pegas yang Telah Dilepas

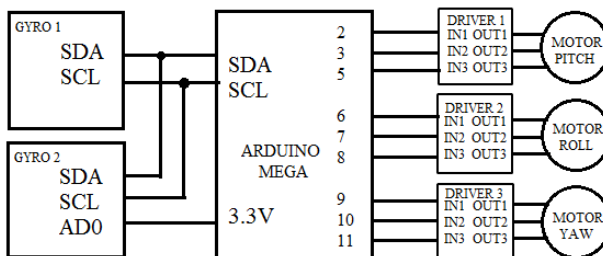
Hasil akhir keseluruhan sistem ditunjukkan pada Gambar 3.8.



Gambar 3.8 Keseluruhan Sistem

Sebagai catatan, kamera yang digunakan pada sistem adalah kamera aksi SJCAM SJ5000X Elite. Kamera ini memiliki *stabilizer* internal sehingga pada penelitian ini pengaturan *stabilizer* dimatikan.

3.1.2 Hubungan Kabel dalam Sistem



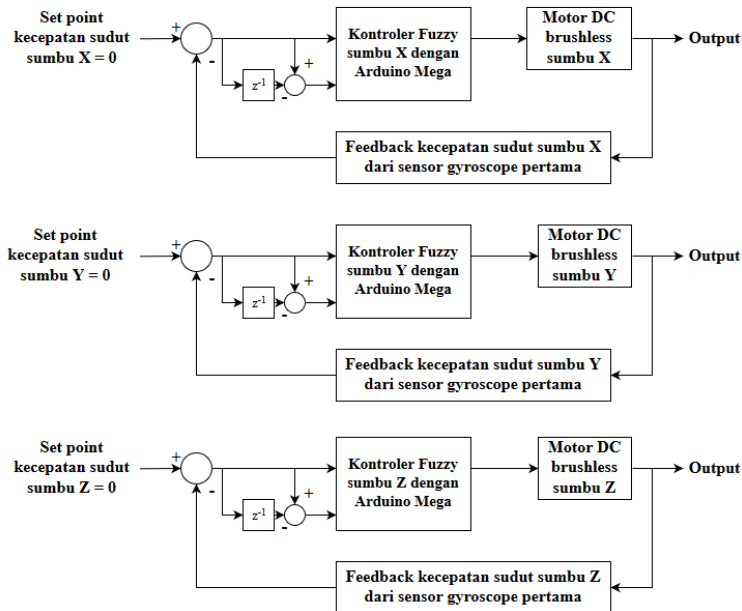
Gambar 3.9 Hubungan Kabel Arduino Mega Dengan Sensor, Motor Driver dan Motor *Brushless* DC

3.2 Perancangan Perangkat Lunak

Pada bagian ini akan dijelaskan mengenai penyusunan perangkat lunak meliputi perangkat lunak untuk mikrokontroler.

3.2.1 Diagram Blok Sistem Metode Pertama

Diagram blok sistem untuk metode pertama adalah sebagai berikut.

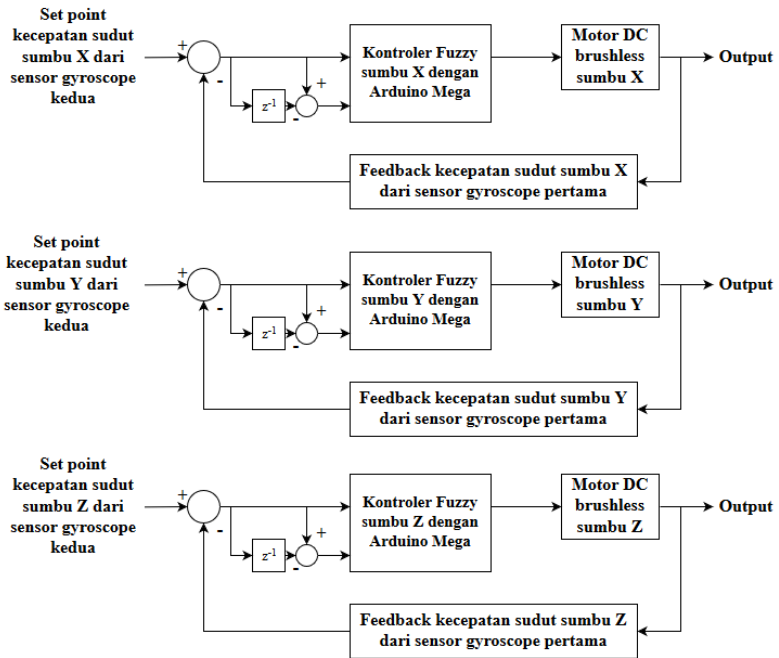


Gambar 3.10 Diagram Blok Sistem Metode Pertama

Nilai *set point* akan dikurangkan dengan nilai *feedback* yang berasal dari *gyroscope* tersebut yang terpasang pada *body* bawah mobil RC. Hasil pengurangan ini akan menjadi sinyal *error*. Masukan dari kontroler Fuzzy yang dirancang adalah sinyal *error* dan sinyal *delta error*. Sinyal *delta error* merupakan sinyal *error* yang sekarang dikurangi sinyal *error* yang sebelumnya. Kemudian sinyal yang keluar dari kontroler adalah sinyal PWM yang berfungsi untuk menggerakkan ketiga motor.

3.2.2 Diagram Blok Sistem Metode Kedua

Diagram blok sistem untuk metode kedua adalah sebagai berikut.



Gambar 3.11 Diagram Blok Sistem Metode Kedua

Sensor *gyroscope* kedua dipasang di bawahudukan kamera berfungsi memberi nilai *set point* yang memiliki besaran derajat/detik. Lalu nilai *set point* ini akan dikurangkan dengan nilai *feedback* yang berasal dari *gyroscope* pertama yang terpasang pada *body* bawah mobil RC. Hasil pengurangan ini akan menjadi sinyal *error*. Masukan dari kontroler Fuzzy yang dirancang adalah sinyal *error* dan sinyal *delta error*. Sinyal *delta error* merupakan sinyal *error* yang sekarang dikurangi sinyal *error* yang sebelumnya. Kemudian sinyal yang keluar dari kontroler adalah besar perubahan nilai PWM yang berfungsi untuk menggerakkan ketiga motor.

3.2.3 Register Clock

MPU-6050 dapat memakai sumber *clock* internal maupun sumber *clock* eksternal yang dapat digunakan. Pilihan untuk internal *clock* adalah MEMS *oscillator* dari sumbu X, sumbu Y, atau sumbu Z *gyroscope*. Sedangkan untuk eksternal *clock* dari 32.768 kHz *square wave*, dan 19.2 MHz *square wave*.

Internal osilator baik digunakan ketika DMP (*Digital Motion Processor*) dimatikan. Sedangkan bila *gyroscope* aktif dianjurkan menggunakan pemilihan sumber clock dari *gyroscope* untuk keakuratan [15].

Pemilihan sumber *clock* untuk MPU-6050 dilakukan dengan mengatur CLKSEL (bit [2:0]) dalam register PWR_MGMT_1. Alamat register PWR_MGMT_1 pada MPU-6050 adalah 0x6B. Pengaturan sumber *clock* untuk MPU-6050 pada register sumber *clock* (PWR_MGMT_1) disajikan dalam Tabel 3.1. Dalam sistem ini, sumber *clock* yang digunakan adalah sumber *clock* internal yang bernilai 8MHz, maka dari itu nilai yang diberikan pada register adalah 0x00.

Tabel 3.1 Pengaturan Register Sumber Clock

Bit ke-			Keterangan
2	1	0	
0	0	0	Internal osilator 8MHz
0	0	1	PLL dengan referensi sumbu X <i>gyroscope</i>
0	1	0	PLL dengan referensi sumbu Y <i>gyroscope</i>
0	1	1	PLL dengan referensi sumbu Z <i>gyroscope</i>
1	0	0	PLL dengan referensi eksternal osilator 32.768 kHz
1	0	1	PLL dengan referensi eksternal osilator 19.2 MHz
1	1	0	Reserved
1	1	1	Clock dimatikan dan time generator dalam kondisi reset

3.2.4 Register Gyroscope

Pengaturan yang dilakukan berkaitan dengan keluaran nilai *gyroscope* dapat diatur pada register GYRO_CONFIG dengan alamat register 0x1B. Struktur dari register GYRO_CONFIG ditunjukkan pada Tabel 3.2

Opsi FS_SEL difungsikan untuk memilih skala penuh yang digunakan pada *gyroscope*. ZG_ST, YG_ST, dan XG_ST digunakan untuk melakukan tes performansi dari masing masing sumbu *gyroscope*. Dalam sistem yang dibuat pengaturan register hanya dilakukan pada FS_SEL.

Terdapat empat pilihan skala penuh pada *gyroscope*. Skala penuh tersebut dapat dipilih dengan memberikan nilai bit ke-0 dan bit ke-1 dari FS_SEL. Nilai pemilihan skala penuh ditunjukkan pada Tabel 3.3 Pada sistem yang dibuat, digunakan skala penuh $\pm 250^\circ/\text{s}$. Oleh karena itu, register konfigurasi *gyroscope* (GYRO_CONFIG) diberikan nilai 0x00.

Tabel 3.2 Register Konfigurasi Gyroscope

Bit ke-	Keterangan
0	-
1	-
2	-
3	FS_SEL[0]
4	FS_SEL[1]
5	ZG_ST
6	YG_ST
7	XG_ST

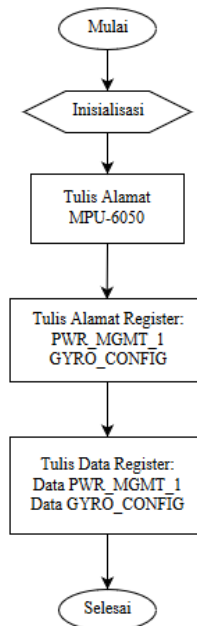
Tabel 3.3 Pengaturan Skala Penuh dan Skala Pemfaktor MPU-6050

FS_SEL bit ke		Skala Penuh ($^\circ/\text{s}$)	Skala Pemfaktor (LSB/($^\circ/\text{s}$))
1	0		
0	0	$\pm 250^\circ$	131
0	1	$\pm 500^\circ$	65.5
1	0	$\pm 1000^\circ$	32.8
1	1	$\pm 2000^\circ$	16.4

3.2.5 Diagram Alir Pengaturan Register Sensor

Pengaturan register pada MPU-6050 dilakukan oleh mikrokontroler dengan komunikasi I2C. Perintah yang digunakan adalah perintah tulis. Dalam pengaturan register MPU-6050 perlu diperhatikan alamat dari MPU-6050 tersebut dan alamat register yang akan diatur. Alamat *default* dari MPU-6050 adalah 0x68, alamat ini dijadikan alamat untuk MPU-6050 pertama sebagai pemberi nilai set poin. Sedangkan alamat dari MPU-6050 kedua sebagai pemberi nilai *feedback* adalah 0x69. Alamat pada MPU-6050 dapat diubah menjadi 0x69 dengan cara memberi tegangan 3.3V pada pin AD0.

Register-register yang akan diatur adalah register *clock*, dan register *gyroscope*. Diagram alir dari proses pengaturan register ditunjukkan pada Gambar 3.12.



Gambar 3.12 Diagram Alir Pengaturan Register Sensor

3.2.6 Pembacaan Data Gyroscope

Pembacaan sensor *gyroscope* pada MPU-6050 disimpan pada register data. Perangkat luar dapat meminta data tersebut dengan menunjuk alamat dari register data tersebut. Data *gyroscope* memiliki lebar data 16-bit yang terdiri dari 8-bit *low byte* dan 8-bit *high byte*. Terdapat tiga buah sumbu pada *gyroscope* sehingga terdapat enam buah register data *gyroscope*.

Karena masing-masing sumbu pada *gyroscope* register datanya terbagi menjadi dua maka untuk mendapatkan data yang valid dari masing-masing sumbu harus dilakukan penggabungan data dari dua buah register tersebut. Ukuran variabel yang disediakan pada program mikrokontroler harus 16-bit bertanda, karena data yang dibaca dapat bernilai positif maupun negatif. Proses penggabungan data dilakukan dengan cara menggeser data *high byte* ke kiri sebesar 8-bit kemudian dijumlahkan dengan data *low byte*. Register data dan alamat *gyroscope* ditunjukkan pada Tabel 3.4.

Tabel 3.4 Register Data *Gyroscope* pada MPU-6050

Nama Register	Alamat	Keterangan
GYRO_XOUT_H	0x43	High byte <i>gyroscope</i> X[15:8]
GYRO_XOUT_L	0x44	Low byte <i>gyroscope</i> X[7:0]
GYRO_YOUT_H	0x45	High byte <i>gyroscope</i> Y[15:8]
GYRO_YOUT_L	0x46	Low byte <i>gyroscope</i> Y[7:0]
GYRO_ZOUT_H	0x47	High byte <i>gyroscope</i> Z[15:8]
GYRO_ZOUT_L	0x48	Low byte <i>gyroscope</i> Z[7:0]

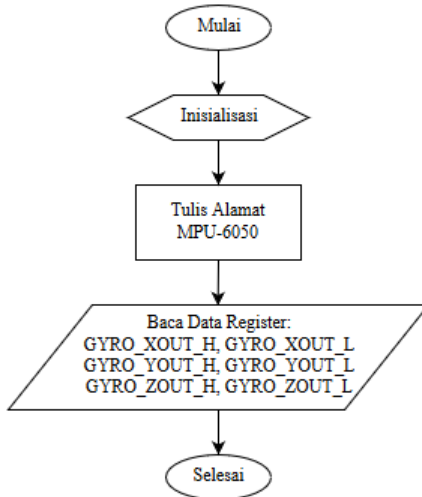
Proses penggabungan data high byte dan low byte ditunjukkan pada Persamaan 3.1.

$$Data\ sensor = (High\ byte \ll 8) + Low\ byte \quad (3.1)$$

di mana *Data sensor* merupakan variabel untuk penyimpanan data sensor *gyroscope* pada setiap sumbu.

3.2.7 Diagram Alir Pembacaan Data Gyroscope

Pembacaan data *gyroscope* dilakukan oleh mikrokontroler. Pembacaan ditujukan pada register data untuk *gyroscope*. Terdapat 6 register data yang dibaca. Diagram alir pembacaan register ditunjukkan pada Gambar 3.13.



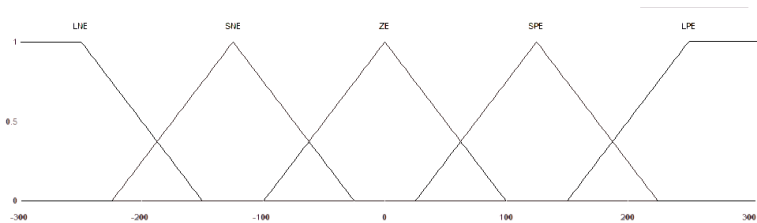
Gambar 3.13 Diagram Alir Pembacaan Data *Gyroscope*

3.2.8 Perancangan Kontrol Fuzzy

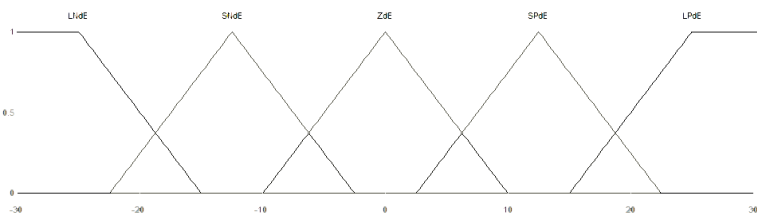
Perancangan kontrol Fuzzy untuk sistem ini adalah menggunakan metode Sugeno. Metode ini dipilih karena ringan untuk dijalankan pada mikrokontroler, perbedaan mendasar dari metode Mamdani adalah, pada metode Sugeno proses defuzzifikasinya menggunakan *Weighted Average*. Hal inilah yang meringankan perhitungan dibandingkan menggunakan metode Mamdani yang defuzzifikasinya menggunakan *Center of Gravity/Centroid*, di mana pada proses perhitungannya memerlukan integral.

Input pertama pada perancangan kontrol menggunakan logika Fuzzy adalah nilai *error* dari penjumlahan nilai kedua sensor *gyroscope* di mana *range* nilainya adalah -2000 sampai 2000. Sedangkan untuk input kedua adalah *delta error*, yakni nilai *error* sekarang dikurangi nilai *error* sebelumnya yang ditetapkan *range*-nya adalah -200 sampai 200.

Membership function untuk error dan delta error adalah sebagai berikut.



Gambar 3.14 *Membership Function Input Error*



Gambar 3.15 *Membership Function Input Delta Error*

Kemudian untuk *rule*-nya disajikan pada tabel berikut.

Tabel 3.5 *Rule Base Sistem Kontrol Fuzzy*

	LNΔE	SNΔE	ZΔE	SPΔE	LPΔE
LNE	LP	LP	MP	SP	SP
SNE	LP	MP	SP	ZM	ZM
ZE	SP	SP	ZM	SN	SN
SPE	ZM	ZM	SN	MN	LN
LPE	SN	SN	MN	LN	LN

Keterangan:

LNE : *Large Negative Error*

SNE : *Small Negative Error*

ZE : *Zero Error*

SPE : *Small Positive Error*

LPE : *Large Positive Error*

LNdE : *Large Negative Delta Error*

SNdE : *Small Negative Delta Error*

ZdE : *Zero Delta Error*

SPdE : *Small Positive Delta Error*

LPdE : *Large Positive Delta Error*

LP : *Large Positive*

MP : *Medium Positive*

SP : *Small Positive*

ZM : *Zero Movement*

SN : *Small Negative*

MN : *Medium Negative*

LN : *Large Negative*

Untuk defuzzifikasi digunakan perhitungan *Weighted Average*.

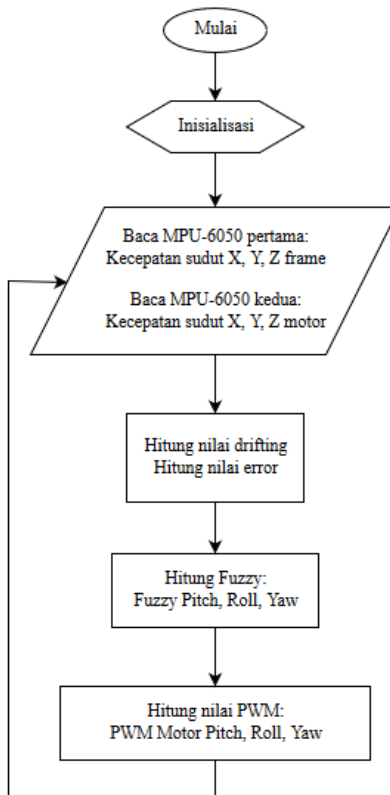
$$Z = \frac{\sum_{i=1}^7 w_i \times \mu(w_i)}{\sum_{i=1}^7 \mu(w_i)} \quad (3.2)$$



Gambar 3.16 Posisi Nilai x Anggota *Output Singleton*

3.2.9 Diagram Alir Program Utama Mikrokontroler

Pada program utama dilakukan proses keseluruhan sistem. Dimulai dari pengaturan register pada MPU-6050, lalu dilanjutkan dengan pembacaan data *gyroscope*, perhitungan kontrol Fuzzy, terakhir menggerakkan motor. Diagram alir dari program utama ditunjukkan pada Gambar 3.17.



Gambar 3.17 Diagram Alir Program Utama Mikrokontroler

3.2.10 Perancangan Program untuk Pengujian Lapangan Menggunakan OpenCV

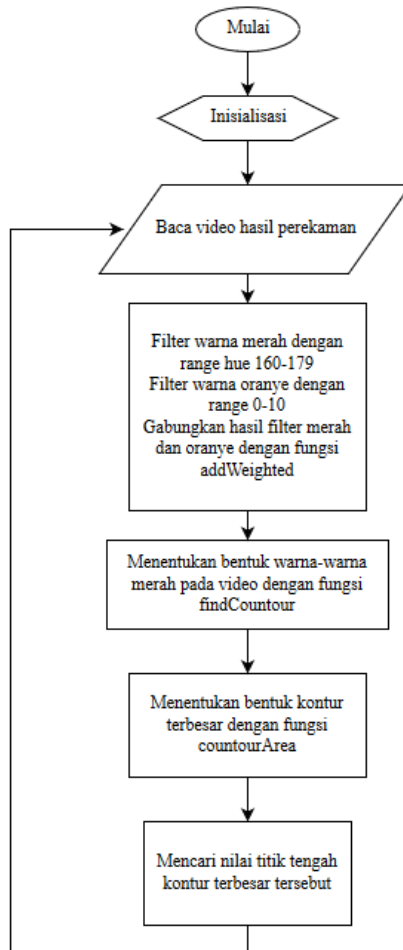
Pada proses pengujian sistem ini melibatkan penginderaan komputer atau pengolahan citra dilakukan menggunakan *library* dari OpenCV. OpenCV adalah suatu *library open source* yang dapat digunakan siapapun tanpa dikenakan biaya. OpenCV ditulis dalam bahasa pemrograman C/C++. OpenCV dikembangkan untuk menambah efisiensi dalam perhitungan penginderaan komputer. OpenCV juga memiliki fokus pengembangan terhadap pengolahan gambar secara langsung dengan sumber dari perangkat seperti kamera. Dengan fasilitas yang ditawarkan OpenCV, OpenCV adalah suatu alat penunjang yang pas untuk metode pengujian lapangan.

Tracking object ini bertujuan untuk pengujian kestabilan pada robot jika dijalankan di jalan yang tidak rata. Pengujian ini memakai cara merekam gambar lingkaran merah pada kertas yang dihadapkan lurus dengan kamera, lalu robot mulai dijalankan lurus. Hasil video kemudian diolah untuk men-tracking lingkaran merah tersebut apakah bergerak sedikit atau bergerak banyak.

Tahap pertama dalam tracking ini adalah menggunakan filter warna yang berdasarkan HSV (*Hue Saturation Value*). Pada pengujian ini akan memfilter warna merah berdasarkan *range hue* warna merah yaitu dari nilai 160-179 dan warna oranye 0-10. Kemudian kedua nilai ini akan digabungkan dengan fungsi *addWeighted*. Tahap kedua adalah menandai bentuk warna-warna merah dalam video yaitu dengan fungsi *findCountour*. Lalu tahap ketiga adalah mencari nilai luasan kontur merah terbesar yang nantinya akan dipilih sebagai objek yang diamati menggunakan fungsi *contourArea*. Selanjutnya kontur tersebut diberi garis kotak di sekelilingnya dengan fungsi *boundingRect*, tahap terakhir yaitu mencari titik tengah berdasarkan garis kotak tersebut. Nilai titik tengah inilah yang nantinya akan diamati dan dibuat grafik, seberapa besar pergeseran objek lingkaran merah tersebut.

3.2.11 Diagram Alir Program untuk Pengujian Lapangan Menggunakan OpenCV

Diagram alir program untuk pengujian hasil video yang direkam dengan sistem ditunjukkan pada Gambar 3.18.



Gambar 3.18 Diagram Alir Program Pengujian Lapangan

Halaman ini sengaja dikosongkan

BAB IV

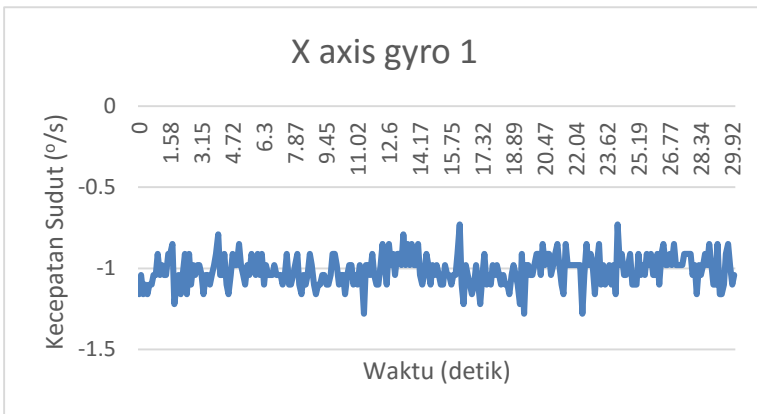
PENGUKURAN DAN ANALISIS SISTEM

Bab ini menjelaskan pengujian sistem sehingga dapat diketahui kinerjanya. Pada bab ini akan dibahas pengujian sensor *gyroscope* dan pengujian kestabilan hasil perekaman video.

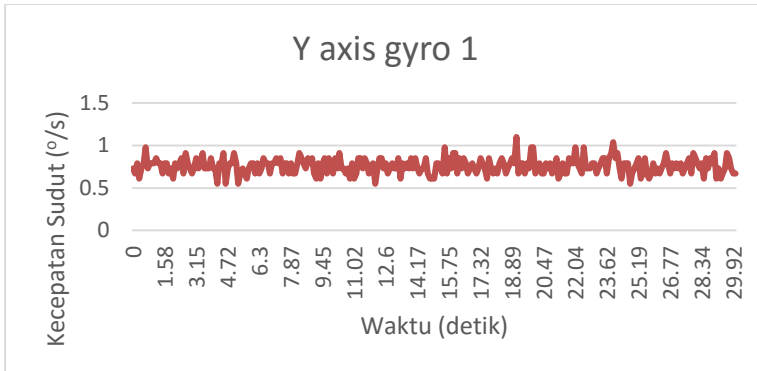
4.1 Pengujian Sensor *Gyroscope*

Pada bagian ini akan diketahui seberapa besar nilai *offset* yang terjadi akibat *drifting* pada masing masing sumbu sensor *gyroscope* pertama dan kedua.

Dari pengujian selama 30 detik dapat diketahui nilai rata-rata offset untuk *gyroscope* pertama sumbu X adalah -1.1 derajat per detik. Dari pengujian selama 30 detik dapat diketahui nilai rata-rata offset untuk *gyroscope* pertama sumbu Y adalah 0.7 derajat per detik.

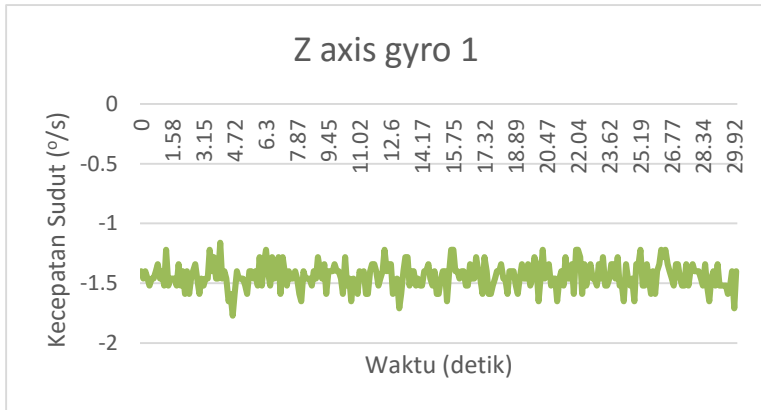


Gambar 4.1 Offset Sumbu X Sensor *Gyroscope* Pertama

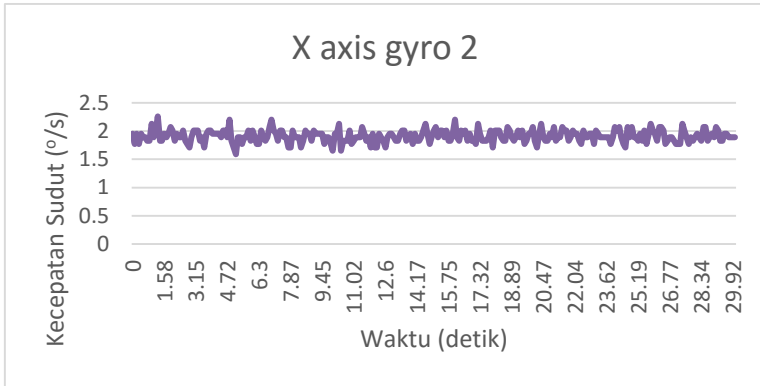


Gambar 4.2 Offset Sumbu Y Sensor *Gyroscope* Pertama

Dari pengujian selama 30 detik dapat diketahui nilai rata-rata offset untuk *gyroscope* pertama sumbu Z adalah -1.4 derajat per detik.

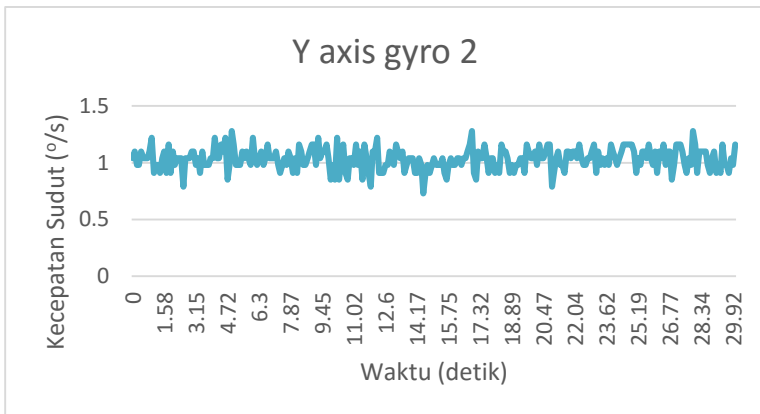


Gambar 4.3 Offset Sumbu Z Sensor *Gyroscope* Pertama

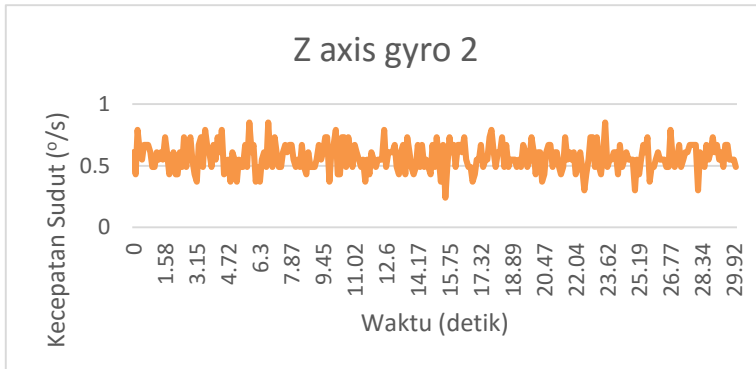


Gambar 4.4 Offset Sumbu X Sensor *Gyroscope* Kedua

Dari pengujian selama 30 detik dapat diketahui nilai rata-rata offset untuk *gyroscope* kedua sumbu X adalah 1.92 derajat per detik. Dari pengujian selama 30 detik dapat diketahui nilai rata-rata offset untuk *gyroscope* kedua sumbu Y adalah 1.1 derajat per detik.



Gambar 4.5 Offset Sumbu Y Sensor *Gyroscope* Kedua

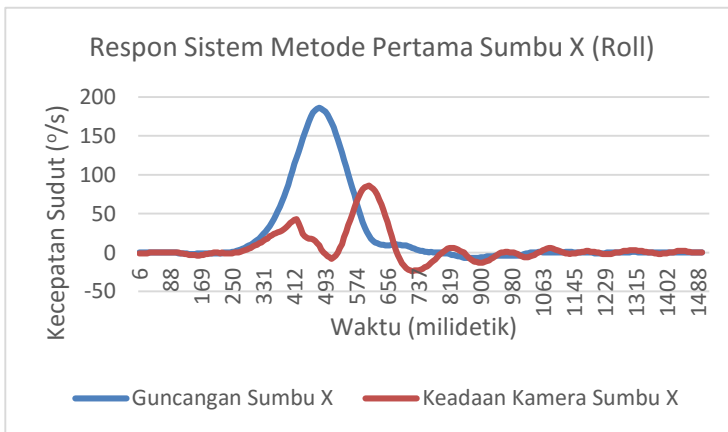


Gambar 4.6 Offset Sumbu Z Sensor *Gyroscope* Kedua

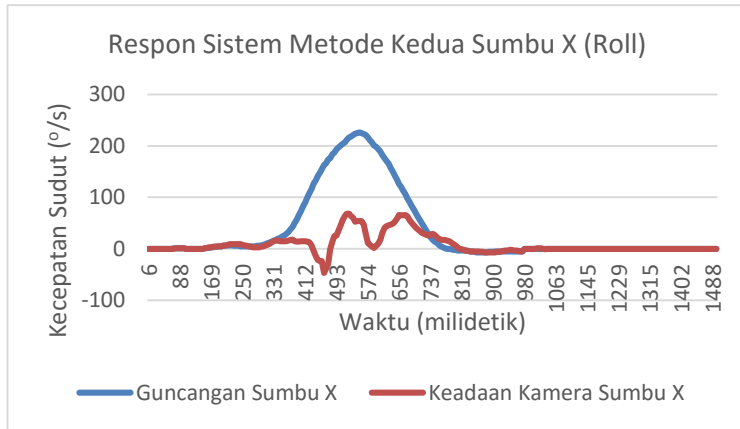
Dari pengujian selama 30 detik dapat diketahui nilai rata-rata offset untuk *gyroscope* kedua sumbu X adalah 0.55 derajat per detik

4.2 Pengujian Kestabilan Sistem dengan Guncangan Terukur

Dari pengujian selama 1500 milidetik dapat terlihat pada grafik bahwa ketika diberi guncangan pada sumbu X, dapat diketahui standar deviasi untuk metode pertama adalah sebesar 20.51. Dari grafik juga terlihat bahwa respon sistem memiliki *overshoot* dan *settling time* yang besar.



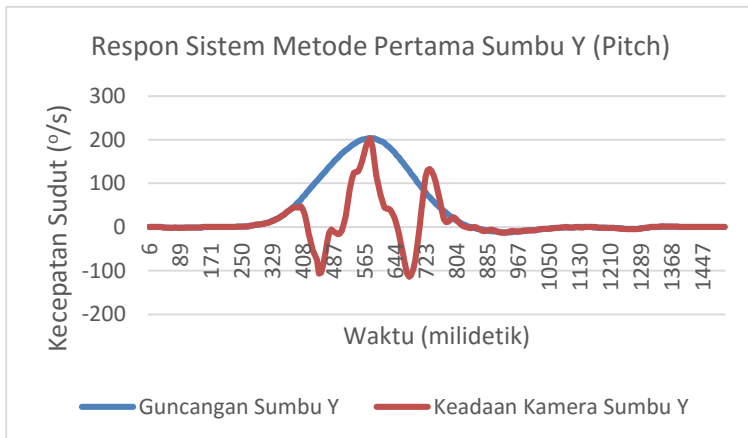
Gambar 4.7 Respon Sistem Sumbu X dengan Metode Pertama



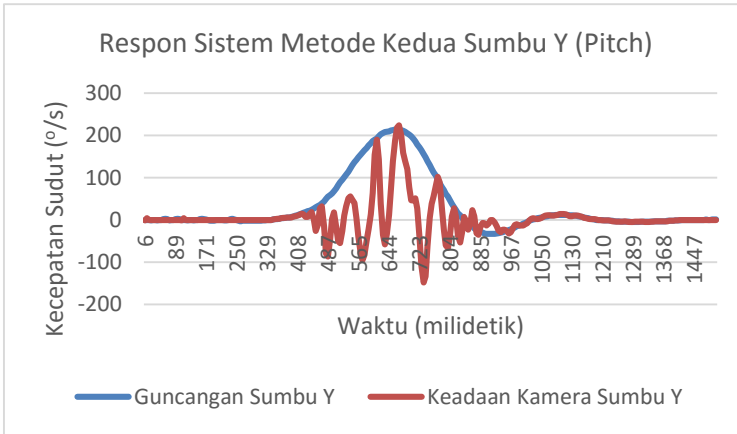
Gambar 4.8 Respon Sistem Sumbu X dengan Metode Kedua

Sementara ketika menggunakan metode kedua, standar deviasinya adalah 18.80. Dari grafik juga terlihat bahwa *overshoot* *settling time* yang besar pada metode pertama dapat dihilangkan.

Ketika diberi guncangan pada sumbu Y, dapat diketahui standar deviasi untuk metode pertama adalah sebesar 45.42. Dari grafik juga terlihat bahwa respon sistem memiliki *overshoot* yang besar.



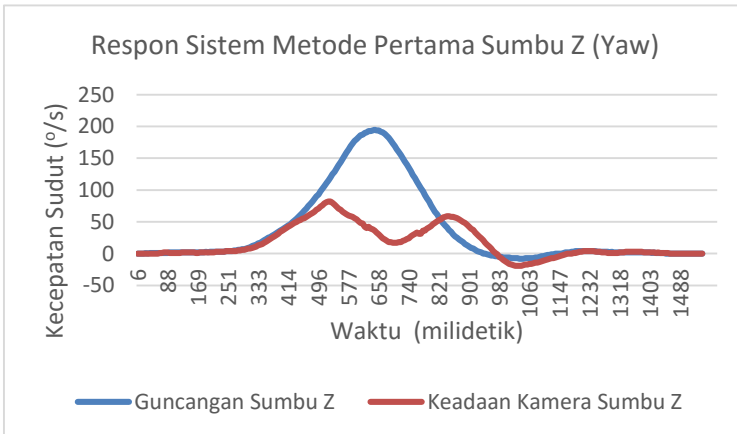
Gambar 4.9 Respon Sistem Sumbu Y dengan Metode Pertama



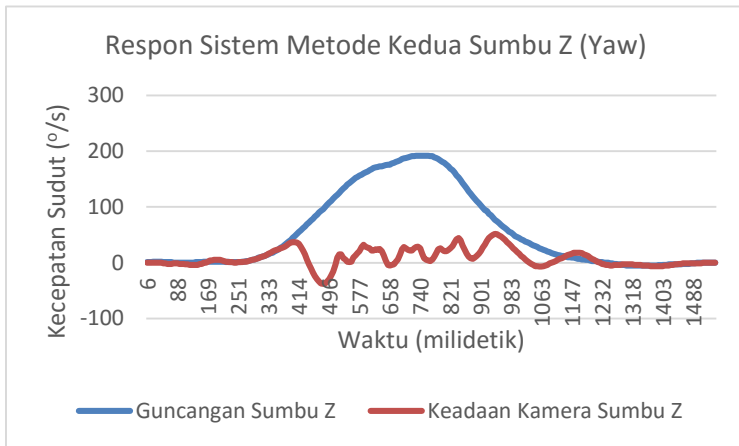
Gambar 4.10 Respon Sistem Sumbu Y dengan Metode Kedua

Sementara ketika menggunakan metode kedua, standar deviasinya adalah 43.82. Dari grafik juga terlihat bahwa *overshoot* yang terjadi pada metode pertama dapat dihilangkan.

Ketika diberi guncangan pada sumbu Z, dapat diketahui standar deviasi untuk metode pertama adalah sebesar 25.16. Dari grafik juga terlihat bahwa respon sistem memiliki *overshoot* yang besar.



Gambar 4.11 Respon Sistem Sumbu Z dengan Metode Pertama



Gambar 4.12 Respon Sistem Sumbu Z dengan Metode Kedua

Sementara ketika menggunakan metode kedua, standar deviasinya adalah 16.01. Dari grafik juga terlihat bahwa *overshoot* yang terjadi pada metode pertama dapat dihilangkan.

Dari grafik dapat dilihat bahwa pada metode pertama memiliki *overshoot* dan *settling time* lebih besar dibanding metode kedua, Sedangkan dari nilai standar deviasi, dapat disimpulkan bahwa metode kedua lebih stabil daripada metode pertama karena metode pertama memiliki standar deviasi yang lebih besar. Dapat disimpulkan bahwa pada sistem dengan kontroler Fuzzy ini, *overshoot* sistem dan *settling time* yang besar masih terjadi jika dibandingkan dengan kontroler PID ketika sama-sama menggunakan satu sensor *gyroscope* [15].

Tabel 4.1 Hasil Pengujian Guncangan Terukur

Sumbu	Standar Deviasi	
	Metode Pertama	Metode Kedua
Sumbu X	20.51	18.80
Sumbu Y	45.42	43.82
Sumbu Z	25.16	16.01

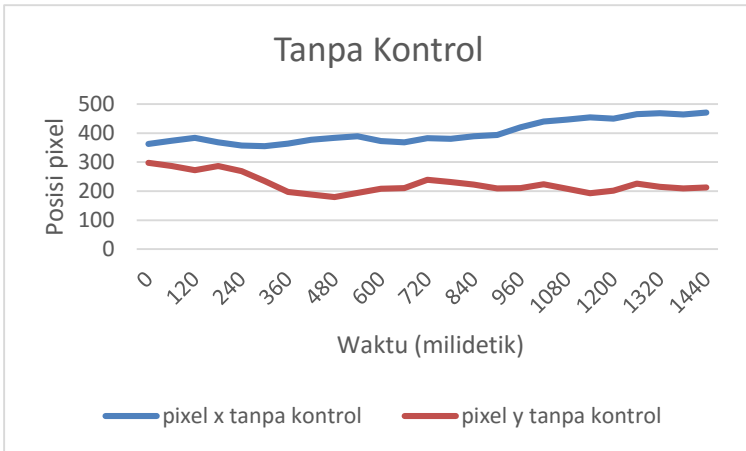
4.3 Pengujian Kestabilan Sistem pada Lapangan

Pada bagian ini akan diketahui seberapa besar pergeseran suatu objek warna terhadap pergerakan mobil RC. Pengujian tanpa menggunakan kontrol dilakukan dengan cara menjepit masing-masing motor pada *frame* menggunakan penjepit kertas. Hasil yang disertakan adalah titik pusat objek berupa nilai *pixel* x dan y dalam video selama 1.5 detik.

Dari data yang disajikan pada grafik di bawah sistem mampu meredam namun tidak cukup baik. Ketika tanpa kontrol perbedaan letak titik tengah objek pada sumbu x dan y memiliki perbedaan lebih dari 100 *pixel*, Hasil ketika tanpa kontrol, standar deviasi *pixel* sumbu x adalah 40.57 dan sumbu y adalah 32.95.

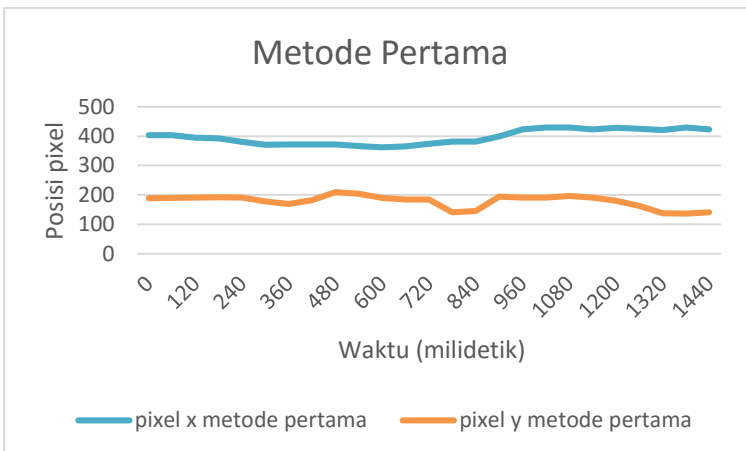


Gambar 4.13 Pengolahan Citra Objek yang Diamati

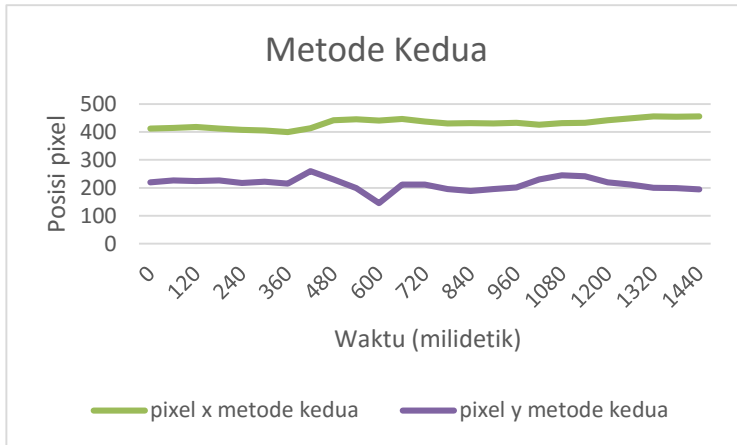


Gambar 4.14 Posisi Perubahan *Pixel* Objek yang Diamati Tanpa Menggunakan Kontrol

Sementara ketika dengan kontrol, perbedaan *pixel* tidak terlalu jauh, sekitar 20 *pixel*. Ketika dengan menggunakan metode pertama, standar deviasi *pixel* sumbu x adalah 24.73 dan sumbu y adalah 21.73.



Gambar 4.15 Posisi Perubahan *Pixel* Objek yang Diamati Dengan Menggunakan Metode Pertama



Gambar 4.16 Posisi Perubahan *Pixel* Objek yang Diamati dengan Menggunakan Metode Kedua

Sementara ketika dengan menggunakan metode kedua, standar deviasi *pixel* sumbu x adalah 16.70 dan sumbu y adalah 22.44. Dari nilai standar deviasi, dapat disimpulkan bahwa metode kedua lebih stabil daripada metode pertama.

Tabel 4.2 Hasil Pengujian Lapangan

Posisi Pixel	Standar Deviasi		
	Tanpa Kontrol	Metode Pertama	Metode Kedua
Pixel X	40.57	24.73	16.70
Pixel Y	32.95	21.73	22.44

LAMPIRAN

Potongan Program pada Arduino Mega

```
void setup() {  
  
    Wire.begin();  
    Wire.beginTransmission(MPU_addr);  
    Wire.write(0x6B); // PWR_MGMT_1 register  
    Wire.write(0x00); // wake up  
    Wire.endTransmission(true);  
  
    Wire.beginTransmission(MPU_addr);  
    Wire.write(0x1B); // GYRO_CONFIG  
    Wire.write(0x00); // select FS_SEL 250 DPS  
    Wire.endTransmission(true);  
  
    Wire.beginTransmission(MPU_addr2);  
    Wire.write(0x6B); // PWR_MGMT_1 register  
    Wire.write(0x00); // wake up  
    Wire.endTransmission(true);  
  
    Wire.beginTransmission(MPU_addr2);  
    Wire.write(0x1B); // GYRO_CONFIG  
    Wire.write(0x00); // select FS_SEL 250 DPS  
    Wire.endTransmission(true);  
  
    //make pwm frequency unaudible  
    TCCR1B = TCCR1B & 0b11111000 | 0x01;  
    TCCR2B = TCCR2B & 0b11111000 | 0x01;  
    TCCR3B = TCCR3B & 0b11111000 | 0x01;  
    TCCR4B = TCCR4B & 0b11111000 | 0x01;  
}  
  
void loop()  
{  
    initsensor();
```

```

calculatevelocity();

pitch = fuzzy(errorY,dErrorY);
roll = fuzzy(errorX,dErrorX);
yaw = fuzzy(errorZ,dErrorZ);

motordrive();
}

void initsensor()
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x43); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,6,true); // request a total of 14
registers
  GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) &
0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) &
0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) &
0x48 (GYRO_ZOUT_L)

  Wire.beginTransmission(MPU_addr2);
  Wire.write(0x43); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr2,6,true); // request a total of 14
registers
  GyX2=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) &
0x44 (GYRO_XOUT_L)
  GyY2=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) &
0x46 (GYRO_YOUT_L)
  GyZ2=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) &
0x48 (GYRO_ZOUT_L)
}

void calculatevelocity()
{

```

```

// apply Gyro scale from datasheet
gsx=GyX/gyroScale;
gsy=GyY/gyroScale;
gsz=GyZ/gyroScale;
gsx2=GyX2/gyroScale;
gsy2=GyY2/gyroScale;
gsz2=GyZ2/gyroScale;

//gyro drift compensation
rx=gsx+1.1;
ry=gsy-0.7;
rz=gsz+1.4;

setpointX=gsx2-1.9;
setpointY=gsy2-1.1;
setpointZ=gsz2-0.55;

//calculate error and delta error
errorX=setpointX+rx; //rx, ry, rz already negative
dErrorX=errorX-errorXold;
errorXold=errorX;

errorY=setpointY+ry;
dErrorY=errorY-errorYold;
errorYold=errorY;

errorZ=setpointZ+rz;
dErrorZ=errorZ-errorZold;
errorZold=errorZ;
}

void motordrive()
{
    incrementpitch = pitch;

    //Check for lookup table overflow and return to opposite end if
    necessary

```

```
pitchStepA = pitchStepA + incrementpitch;  
if(pitchStepA > ArraySize) pitchStepA = pitchStepA-ArraySize;  
if(pitchStepA < 0) pitchStepA = ArraySize;
```

```
pitchStepB = pitchStepA+phaseShiftB;  
if(pitchStepB > ArraySize) pitchStepB = pitchStepB-ArraySize;  
if(pitchStepB < 0) pitchStepB = ArraySize;
```

```
pitchStepC = pitchStepA+phaseShiftC;  
if(pitchStepC > ArraySize) pitchStepC = pitchStepC-ArraySize;  
if(pitchStepC < 0) pitchStepC = ArraySize;
```

```
analogWrite(pitchMotor1, pwmSin[pitchStepA]);  
analogWrite(pitchMotor2, pwmSin[pitchStepB]);  
analogWrite(pitchMotor3, pwmSin[pitchStepC]);
```

```
incrementroll = roll;
```

```
//Check for lookup table overflow and return to opposite end if  
necessary
```

```
rollStepA = rollStepA + incrementroll;  
if(rollStepA > ArraySize) rollStepA = rollStepA-ArraySize;  
if(rollStepA < 0) rollStepA = ArraySize;
```

```
rollStepB = rollStepA+phaseShiftB;  
if(rollStepB > ArraySize) rollStepB = rollStepB-ArraySize;  
if(rollStepB < 0) rollStepB = ArraySize;
```

```
rollStepC = rollStepA+phaseShiftC;  
if(rollStepC > ArraySize) rollStepC = rollStepC-ArraySize;  
if(rollStepC < 0) rollStepC = ArraySize;
```

```
analogWrite(rollMotor1, pwmSin[rollStepA]);  
analogWrite(rollMotor2, pwmSin[rollStepB]);  
analogWrite(rollMotor3, pwmSin[rollStepC]);
```

```
incrementyaw = yaw;
```

```
//Check for lookup table overflow and return to opposite end if
necessary
```

```
  yawStepA = yawStepA + incrementyaw;
  if(yawStepA > ArraySize) yawStepA = yawStepA-ArraySize;
  if(yawStepA < 0) yawStepA = ArraySize;
```

```
  yawStepB = yawStepA+phaseShiftB;
  if(yawStepB > ArraySize) yawStepB = yawStepB-ArraySize;
  if(yawStepB < 0) yawStepB = ArraySize;
```

```
  yawStepC = yawStepA+phaseShiftC;
  if(yawStepC > ArraySize) yawStepC = yawStepC-ArraySize;
  if(yawStepC < 0) yawStepC = ArraySize;
```

```
  analogWrite(yawMotor1, pwmSin[yawStepA]);
  analogWrite(yawMotor2, pwmSin[yawStepB]);
  analogWrite(yawMotor3, pwmSin[yawStepC]);
}
```

```
float triangle (float x, float g, float h)
```

```
{
  float result;
  if ((x <= (g - h / 2) || x >= (g + h / 2)) && h > 0 ){
    result = 0;
  }
  else if ((x > (g - h / 2) && x < (g + h / 2)) && h > 0) {
    result = 1 - 2 * abs(x - g) / h;
  }
  else if (x == g && h == 0){
    result = 1;
  }
  return result;
}
```

```
float trapezoid (float x, float c, float d, float e, float f)
```

```
{
  float result;
```



```

if ((x < (c - d) && d > 0) || (x > (e + f) && f > 0)){
    result = 0;
}
else if (x < c){
    result = (x - c + d) / d;
}
else if (x < e || (x == c && d == 0) || (x == e && f == 0)){
    result = 1;
}
else if (x <= (e + f)) {
    result = (e + f - x) / f;
}
return result;
}

```

```

float fuzzy(float x, float y)
{
    //membership function
    float LNEr = trapezoid(x, -300, 0, -250, 100);
    float SNEr = triangle(x, -125, 200);
    float ZEr = triangle(x, 0, 200);
    float SPEr = triangle(x, 125, 200);
    float LPEr = trapezoid(x, 250, 100, 300, 0);

    float LNEr = trapezoid(y, -300, 0, -25, 10);
    float SNdEr = triangle(y, -12.5, 20);
    float ZdEr = triangle(y, 0, 20);
    float SPdEr = triangle(y, 12.5, 20);
    float LPdEr = trapezoid(y, 25, 10, 300, 0);

    //rule
    float x1 = min(LNEr,LNdEr);
    float x2 = min(LNEr,SNdEr);
    float x3 = min(LNEr,ZdEr);
    float x4 = min(LNEr,SPdEr);
    float x5 = min(LNEr,LPdEr);
    float x6 = min(SNEr,LNdEr);
    float x7 = min(SNEr,SNdEr);
}

```

```

float x8 = min(SNEr,ZdEr);
float x9 = min(SNEr,SPdEr);
float x10 = min(SNEr,LPdEr);
float x11 = min(ZEr,LNdEr);
float x12 = min(ZEr,SNdEr);
float x13 = min(ZEr,ZdEr);
float x14 = min(ZEr,SPdEr);
float x15 = min(ZEr,LPdEr);
float x16 = min(SPEr,LNdEr);
float x17 = min(SPEr,SNdEr);
float x18 = min(SPEr,ZdEr);
float x19 = min(SPEr,SPdEr);
float x20 = min(SPEr,LPdEr);
float x21 = min(LPEr,LNdEr);
float x22 = min(LPEr,SNdEr);
float x23 = min(LPEr,ZdEr);
float x24 = min(LPEr,SPdEr);
float x25 = min(LPEr,LPdEr);

```

```

float y7a = max(x1,x2);
float y7 = max(y7a,x6);

```

```

float y6 = max(x3,x7);

```

```

float y5a = max(x4,x5);
float y5b = max(y5a,x8);
float y5c = max(y5b,x11);
float y5 = max(y5c,x12);

```

```

float y4a = max(x9,x10);
float y4b = max(y4a,x13);
float y4c = max(y4b,x16);
float y4 = max(y4c,x17);

```

```

float y3a = max(x14,x15);
float y3b = max (y3a,x18);
float y3c = max (y3b,x21);
float y3 = max(y3c,x22);

```

```

float y2 = max(x19,x23);

float y1a = max(x20,x24);
float y1 = max(y1a,x25);

//defuzzification
float numerator=((y1*-25)+(y2*-20)+(y3*-
10)+(y4*0)+(y5*10)+(y6*20)+(y7*25));
float denominator=(y1+y2+y3+y4+y5+y6+y7);

float z = numerator/denominator;
return z;
}

```

Program untuk Pengujian Menggunakan OpenCV

```

#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>
#include <Windows.h>

using namespace cv;
using namespace std;

int main(int, char**)
{
    VideoCapture cap("E:/VID_20160518_161245.mp4"); //
    open the default camera

    if (!cap.isOpened()) // check if we succeeded
        return -1;

    while(1)
    {
        Mat frame;
        cap >> frame; // get a new frame from camera
    }
}

```

```

    bool success = cap.read(frame);
    if (!success){
        cout << "Video stopped" << endl;
        Sleep(10000);
        break;
    }

    Mat hsv, lower, upper, mask;
    Rect bounding_rect;

    vector<vector<Point>> contours; // Vector for
storing contour
    vector<Vec4i> hierarchy;

    double largest_area = 0;
    int largest_contour_index = 0;

    int cx = 0;
    int cy = 0;

    cvtColor(frame, hsv, CV_BGR2HSV);
    cv::inRange(hsv, Scalar(0, 100, 100), Scalar(10,
255, 255), lower);
    cv::inRange(hsv, Scalar(160, 100, 100), Scalar(179,
255, 255), upper);
    addWeighted(lower, 1, upper, 1, 0, mask);

    findContours(mask, contours, hierarchy,
CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE); // Find the
contours in the image

    for (size_t i = 0; i < contours.size(); i++) // iterate
through each contour.
    {

        double a = contourArea(contours[i], false);
// Find the area of contour
        if (a > largest_area){

```

BAB V

PENUTUP

5.1 Kesimpulan

Dari seluruh nilai standar deviasi yang didapatkan pada pengujian, dapat disimpulkan bahwa metode kedua lebih stabil daripada metode pertama karena nilai standar deviasi metode kedua lebih kecil 1.71 s.d. 9.15 daripada metode pertama. Selisih standar deviasi pada pengujian perekaman video antara tanpa kontrol dan menggunakan metode pertama adalah 15.84 untuk *pixel x*, dan 11.22 untuk *pixel y*, dan selisih antara tanpa kontrol dan menggunakan metode kedua adalah 23.87 untuk *pixel x*, dan 10.55 untuk *pixel y*. Dapat disimpulkan bahwa sistem yang dibuat ini dengan kontroler Fuzzy ini, *overshoot* sistem dan *settling time* yang besar masih terjadi jika dibandingkan dengan kontroler PID ketika sama-sama menggunakan satu sensor *gyroscope*.

5.2 Saran

Setelah melakukan pengujian sistem, terdapat beberapa saran untuk pengembangan sistem supaya bisa menjadi lebih baik, yaitu perlunya merancang kembali kontrol Fuzzy agar hasil yang didapat lebih baik. Masalah osilasi pada sumbu Y, perlu dilakukan pengujian motor *brushless* yang memiliki torsi yang lebih besar, momen inersia beban (kamera) cukup mempengaruhi osilasi yang terjadi.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

- [1] M. Nur Shobakh, “Pengembangan Robot Pengikut Garis Berbasis Mikrokontroler Sebagai Meja Pengantar Makanan Otomatis”, ITS Surabaya, 2012.
- [2] <URL: <http://theboredengineers.com/2012/05/the-quadcopter-basics/>> 23 Mei 2016
- [3] <URL: <http://stta.ac.id/blog/?p=131>> 23 Mei 2016
- [4] <URL: <https://www.iotcusa.net/br484405>> 23 Mei 2016
- [5] Barker Keith L., “Stability Boundaries for Systems With Frequency-Model Feedback and Complementary Filter”, NASA Trchnical Paper 1744, November 1980
- [6] <URL: <http://playground.arduino.cc/Main/MPU-6050>> 21 Mei 2016
- [7] Campa R., Torres E., Salas F., Santibanez V., “On Modelling and Parameter Estimation of *Brushless* DC Servoactuators for Position Control Tasks”, Proceedings of the 17th World Congress, The International Federation of Automatic Control, Seoul, Korea, July 2008.
- [8] Gabriel D.L., Meyer J., Plessis F., “*Brushless* DC Motor Characterisation and Selection for a Fixed Wing UAV”, IEEE Africon 2011.
- [9] <URL: <http://www.berryjam.eu/2015/04/driving-bldc-gimbals-at-super-slow-speeds-with-arduino/>> 23 Mei 2016
- [10] <URL: <http://old.fortronic.it/user/file/A%26VElettronica/Brushless-DC-Motors---Part-I%26II.pdf>> 22 Mei 2016
- [11] <URL: <http://www.baixonline.net/projecte/memoria.pdf>> 21 Mei 2016
- [12] Jannus Maurits Nainggolan, “Logika *Fuzzy* (*Fuzzy Logic*) : Teori dan Penerapan Pada Sistem Daya (Kajian Pengaruh Induksi Medan Magnet)”, <URL: <http://member.unila.ac.id/~ftelektro/lab/ltp/dokumen/Fuzzy%20Logic%20Paper.doc>> 23 Mei 2016
- [13] <URL: http://k12008.widyagama.ac.id/ai/diktatpdf/Logika_Fuzzy.pdf> 23 Mei 2016
- [14] <URL: https://www.teledynedalsa.com/public/corp/CCD_vs_CMOS_Litwiller_2005.pdf> 07 Juni 2016

- [15] Jodit Sulisty, “Desain dan Implementasi Gimbal Menggunakan Brushless Motor untuk Stabilisator Posisi Kamera”, ITS Surabaya, 2014.

BIOGRAFI PENULIS



Fahrezi Alwi Muhammad dilahirkan di Jakarta pada 28 Maret 1995. Penulis memulai jenjang pendidikan di SDN Pejaten Timur 03, setelah lulus SD tahun 2006 penulis melanjutkan ke SMPN 107 Jakarta, lulus SMP pada tahun 2009, penulis kemudian melanjutkan ke SMAN 38 Jakarta, setelah lulus SMA pada tahun 2012 penulis melanjutkan studi di Institut Teknologi Sepuluh Nopember jurusan Teknik Elektro dengan konsentrasi di bidang studi Elektronika.

Penulis dapat dihubungi melalui alamat email fahrezialwi@gmail.com.